CLEAN WATER WATER POLLUTION CONTROL RESEARCH SERIES • 16110 FIO 01/72

W72-09228

PB 209 893

Use of General Equilibrium in Regional Water Resource Planning

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, VA 22151

U.S. ENVIRONMENTAL PROTECTION AGENCY

165

WATER POLLUTION CONTROL RESEARCH SERIES

The Water Pollution Control Research Series describes the results and progress in the control and abatement of pollution in our Nation's waters. They provide a central source of information on the research, development, and demonstration activities in the Environmental Protection Agency, through inhouse research and grants and contracts with Federal, State, and local agencies, research institutions, and industrial organizations.

Inquiries pertaining to Water Pollution Control Research Reports should be directed to the Chief, Publications Branch, Research Information Division, Research and Monitoring, Environmental Protection Agency, Washington, D. C. 20460.

SELECTED WATER RESOURCES ABSTRACTS
INPUT TRANSACTION FORM

W 72-09228

USE OF GENERAL EQUILIBRIUM IN REGIONAL WATER RESOURCE PLANNING,

~~Poirier, J. E., and Harrington, J. M., Jr.~~

Georgetown University, Washington, D.C.

EPA 16110 FOS 01/72

Water Pollution Control Research Series

~~Prepared for: The Office of Research and Monitoring~~, ~~U.S.~~ Environmental Protection Agency, ~~Project #16110 FIO, p 1-162,~~ January, 1972. 2 fig, 3 tab, 2 append, 11 ref.

General equilibrium analysis is shown to be a feasible tool for estimating the optimal level of public goods in a regional economy and the optimal allocation of public funds to obtain the desired level. This analysis provides a methodology for investigating the externalities associated with various forms of production. An interaction or trade mechanism is presented which will force a regional economy into equilibrium with the economy in which it is embedded: Relative prices will be identical in these economies for their common commodities. A technique is presented by which all public goods can be treated in a general equilibrium framework. The results of some numeric computations indicate how the methodology works with a regional economy which has a public good. Since one of the conclusions from a general equilibrium application is the optimum supply of any commodity, this technique can be used to determine efficient and economical methods of controlling water pollution in a given region. (Bell-Cornell)

*Algorithms, *Optimization, *Estimating, *Regional economics, *Prices, *Regional analysis, *Supply, *Resource allocation, *Water pollution control, *Water resources, Computer programs.

*Equilibrium analysis, *Public goods, *Public funds, *Externalities, *Shadow prices.

Copy available from GPO Sup Doc as ~~as~~ , $1.50 ; microfiche from NTIS as PB , $0.95.

06A, 06B

PB-209 893

Send To:
WATER RESOURCES SCIENTIFIC INFORMATION CENTER
U S DEPARTMENT OF THE INTERIOR
WASHINGTON D C 20240

Janet Bell | Cornell

USE OF GENERAL EQUILIBRIUM IN REGIONAL WATER RESOURCE PLANNING

by

Georgetown University
Washington, D.C. 20007

for the

Office of Research and Monitoring

ENVIRONMENTAL PROTECTION AGENCY

Project #16110 FIO

January 1972

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402 - Price $1.50

EPA Review Notice

This report has been reviewed by the Environmental Protection Agency and approved for publication. Approval does not signify that the contents necessarily reflect the views and policies of the Environmental Protection Agency, nor does mention of trade names or commercial products constitute endorsement or recommendation for use.

ABSTRACT

This report presents a methodology for applying general equilibrium analysis to a regional economy. An interaction or trade mechanism is presented which will force a regional economy into equilibrium with the economy in which it is embedded in the sense that the relative prices will be identical in these economies for their common commodities. Specifically one or more consumers in the regional economy is assumed to hold money resources which are allocated to the purchase (sale) in the surrounding economy according to the plan determined by the algorithm presented in this report.

The assumptions required for applying general equilibrium analysis to a region are stated. A technique is presented by which public goods can be treated in a general equilibrium framework. Since one of the conclusions from a general equilibrium application is the optimum supply of any commodity, this technique can determine the optimum supply for a region for such public goods as clean water (air, etc.). There is an equilibrium price for a public good for each consumer and these different prices determine what each consumer should be taxed for the public good.

The results of some numeric computations are presented indicating how the use of the algorithm works with a regional economy which has a public good.

This report was submitted in fulfillment of Project Number 16110 FIO under the sponsorship of the Federal Water Pollution Control Administration.

CONTENTS

Section		Page
I	Conclusions	1
II	Recommendations	3
III	Introduction	5
IV	General Equilibrium Analysis in a Regional Economy	9
	The General Equilibrium Model	
	The Representation of a Public Good in General Equilibrium Analysis	
	Allocation of Money Resources to a Regional Economy	
	A Numerical Example	
V	The Algorithm of the Fundamental Theorem	49
	The Fundamental Theorem	
	Application of the Fundamental Theorem to the General Equilibrium Model	
	Application of the Fundamental Theorem to the Allocation Model	
	An Alternative Method of Solution of the Allocation Model	
VI	Acknowledgments	113
VII	References	115
VIII	Appendices	117

FIGURES

		PAGE
1	The set P formulated algebraically for the case of $n=3$, $G=5$, $Q_1=0$, $Q_2=1$, and $Q_3=-1$	51
2	A geometric view of the mapping from a set P_k of allocations to the unit simplex of prices relative to some hypothetical allocation model	104

TABLES

No.		Page
1	Specification of a Hypothetical Regional Economy with a Public Good	38
2	Results of Allocation Model Applied to the Economy of Table 1 under the Assumption that Only the Factors of Production Can Be Traded	41
3	Results of Allocation Model Applied to the Economy of Table 1 under the Assumption that Unskilled Labor is Localized to the Regional Economy and the Locally Produced Good Can Be Traded	43

SECTION I

CONCLUSIONS

1. General Equilibrium analysis appears to be a feasible tool for investigating the optimal level of public goods in a regional economy and the optimal allocation of public funds to obtain the desired level.

2. In as much as pollution can be interpreted as detracting from a public good (water, air, etc.), this analysis provides a methodology for investigating the externalities associated with various forms of production.

3. This analysis provides a method which can determine the optimum method of controlling water (air, etc.) pollution in a region. Specifically it could determine whether pollution should be controlled by requiring the producer to adopt a new (and presumably more costly) technology or whether the existing technology should be retained and methods of cleaning up after the pollution be adopted.

SECTION II

RECOMMENDATIONS

1. Appropriate data should be obtained so that the techniques described in this paper could be applied to a realistic situation. It is only in this way that the real applicability of this analysis can be tested.

2. Estimates of utility and/or demand functions are crucial to this analysis. Further investigations of methods of estimation and their applicability should be pursued.

3. Alternative computational techniques should be investigated to increase the efficiency of the algorithm. One such technique is to search only the polar cone of prices in the general equilibrium solution.

4. If pollutants are treated as "bads" with costly disposal, their prices will be negative. Negative prices should be investigated to see if they provide a more realistic approach to the treating of pollution of public goods and show more clearly the externalities of production with pollution.

Preceding page blank

SECTION III

INTRODUCTION

The possibility of computing a set of general equilibrium prices for an economy with fairly restrictive yet standard specifications has only recently been achieved by Herbert Scarf. Such a model does not at first appear relevant to applied economics since it requires the specification of all the interlocking parts of an economy. If some segment of an economy were isolated for analysis, there would be no guarantee that this segment would be in equilibrium with the rest of the economy. In fact, for a developed economy, general equilibrium would have to apply to a whole country if foreign trade were ignored, or to the whole world if trade between countries were taken into consideration. Under these conditions the task of specifying meaningfully even the major parts of an economy is simply prohibitive.

While examining the possibility of applying general equilibrium analysis to regional water resource planning, it became increasingly clear that if some appropriate form of interaction could be specified between a regional economy and a larger economy in which it is embedded, it would then be possible to simplify drastically the specification requirements for general equilibrium analysis. Any regional or small economy for which it would be feasible to specify the interlocking parts required for general equilibrium analysis would then be artificially bounded by the rest of the economy at least for purposes of analysis. Assuming that the larger economy is already in

Preceding page blank

some form of general equilibrium, then all its prices can be considered as general equilibrium prices. To be in equilibrium with the larger economy, the regional or smaller economy must then reach the same set of equilibrium prices for those commodities common to both economies. What is proposed in the present study is an adjustment mechanism which brings the regional economy as close to equilibrium with the larger economy as is consistent with the given specification of the regional economy.

The adjustment mechanism consists in allowing a net inflow of commodities from the larger economy into the regional economy whose value is fixed in terms of the prices of the larger economy. The source of such money resources can be government subsidies, foreign aid, or any economic source of liquid funds private or public. The term net inflow is used to indicate that the adjustment mechanism also permits the sale of commodities from the regional to the larger economy. The adjustment mechanism determines the optimum allocation of the money resources to the commodities which can be traded between the two economies, so that their relative prices in the smaller economy become as close as possible to their relative prices in the larger economy.

The remarkable fact which the present study proceeds to demonstrate is that precisely the same algorithm that computes general equilibrium prices can be used to compute the optimum allocation of the money resources. This algorithm is embedded in the proof of what will be called the Fundamental Theorem.

Section IV of this report will introduce the model and the assumptions behind the model. General Equilibrium analysis will be presented first, followed by the treatment of public goods in a general equilibrium framework. The methodology for isolating a regional economy for analysis with the tools of general equilibrium will then be presented as the allocation model. Section IV will conclude with the model applied to a hypothetical regional economy. The purpose of this model is to demonstrate the workings and possible results of the model and not to present a realistic application.

A regional economy which contains a public good lends itself ideally to the allocation model. Indeed, since there can be no market price associated with a public good, the optimum supply of such a commodity must be approached from a purely theoretical basis. When a region is forced into equilibrium with the economy in which it is embedded, the general equilibrium solution produces a set of shadow prices of the public good, one price for each consumer, at which the supply of this public good is optimized subject to the given specification of the regional economy.

Section V will present in detail the theory justifying the solutions to the models presented in Section IV. Specifically the description and proof of the algorithm will be presented as the Fundamental Theorem. This theorem will then be applied to the general equilibrium model to solve for a set of equilibrium prices and also to the allocation model to determine the optimum allocation of money resources to a region so that the region will be in equilibrium with the economy in which it is embedded.

With this methodology, it is hoped that General Equilibrium analysis can become a convenient tool for applied economics. As the numerical example shows, actual computations of equilibrium solutions can be obtained, and comparative studies by parametric changes in the specification of the regional economy are made possible with this methodology. All of the models of this study have been programmed and the details of the program can be found in Appendix B.

SECTION IV

GENERAL EQUILIBRIUM ANALYSIS IN A REGIONAL ECONOMY

The use of general equilibrium analysis in applied economics is limited by the fact that most economies are so large that the specification requirements are prohibitive. This section will present a methodology which makes the analysis of a regional economy with the tools of general equilibrium analysis feasible. The assumptions of the general equilibrium model will be presented first, followed by a description of a technique for treating public goods in a general equilibrium framework. The methodology for applying general equilibrium in a regional economy will follow and a hypothetical numerical example will be presented which demonstrates the feasibility of this analysis.

The General Equilibrium Model

General equilibrium theory postulates an economy with a finite number of consumers and producers who own, exchange, and transform a finite number of physical input and output commodities. Each consumer is assumed to have a well defined system of preferences for any set of physical commodities which could be offered to him in the sense that he can rank all such commodity bundles in a consistent way. It can be assumed that such a preference ordering could be represented mathematically in the form of a utility function from which demand functions can be derived specifying the quantity of every good which will be demanded at any set of prices once the consumer's income is

given. The derivations of the demand functions are made under the assumption of perfect competition so that each consumer is a price taker and his choices do not affect individually the price structure. As a practical matter all that is needed are demand functions which may in many cases be easier to estimate than the utility functions from which they are derived.

For this paper we will assume that each consumer has a specified preference ordering represented by a utility function of either the activity, Cobb-Douglas, or C.E.S. type. Each consumer is furthermore assumed to have a fixed bundle of commodities w at the beginning of the analysis from which the consumer derives his income. Specifically, if p is the set of prices prevailing in the economy and there are n commodities, $\sum_{k=1}^{n} p_k w_k$ represents the income of the consumer.

In the case where the consumer's preference ordering is represented by a utility function of the activity type, the demands will always be in fixed proportions. Specifically if b_j is the element in the activity vector associated with commodity j, the demand for commodity j will be

$$x_j = qb_j \quad \text{where} \quad q = \frac{\sum_k p_k w_k}{\sum_k p_k b_k}$$

This demand function is homogeneous of degree zero in the prices and satisfies the consumers budget constraint. This type of ordering

reflects the attitude of a consumer who is confronted with goods which are all perfect complements to each other, an ordering which could very well apply to a government having a fixed set of priorities which must be carried out at all cost.

A second type of representation of a consumer's preference ordering is a Cobb-Douglas type utility function. In this case a consumer's utility for a commodity bundle x, $U(x)$ can be represented as follows:

$$U(x) = A \prod_k (x_k)^{b_k} \quad \text{where} \quad \sum_k b_k = 1$$

Such a function is linear homogeneous and gives rise to a single valued demand function for any commodity j for any price system p. This demand function is derived in Appendix A and is of the following form:

$$x_j = b_j (\sum_k p_k w_k)/p_j$$

provided $p_j \neq 0$, in which case x_j is a free good which does not appear on the market. This function is homogeneous of degree zero in the prices as is required of all demand function. The Cobb-Douglas utility function is the limiting case of the C.E.S. utility function when the elasticity coefficient is equal to unity. An interesting property of this type of utility function is that a consumer who orders his preferences accordingly, spends a fixed proportion of his income on each commodity no matter which price system prevails in the economy. Specifically he spends b_j of his total income on commodity j.

Perhaps a more realistic mathematical representation of a consumer's preference ordering is given by a function of the C.E.S. (constant elasticity of substitution) type. This may be written as follows:

$$U(x) = \left(\sum_k (b_k)^{1-a} (x_k)^a \right)^{1/a}$$

where $e = 1/(1-a)$ is the constant elasticity of substitution between the input commodities. This function is linear homogeneous and gives rise to a single valued demand function for any commodity j for any price system p. This demand function is derived in Appendix A and is of the following form:

$$x_j = \frac{b_j \sum_k p_k w_k}{(p_j)^e \sum_k b_k (p_k)^{1-e}}$$

provided $p_j \neq 0$ in which case x_j is a free good which does not appear on the market. This function is again homogeneous of degree zero in the prices.

The fact that all the consumer demand functions are homogeneous of degree zero in the prices means that only relative prices need be considered. Indeed, any scale of the price system would affect income as well as the purchase prices, and the consumers would have identical demands. Specifically we will assume that all the prices have been scaled to add up to unity. It should be noted that by construction all of the elements in the consumer demand function will be nonnegative.

In applying general equilibrium analysis to the real world one need not have available the preference ordering for every consumer. If one assumes a homogeneity of preferences among consumer groups, such as unskilled labor, professional labor, etc., one may use these consumer groups as consumers. This simplification is probably necessary in any real application.

The most difficult data to obtain in applying general equilibrium analysis is clearly the parameters of these utility (or demand) functions. Particularly with regard to public goods, actual demand can never be estimated directly as there is no market price for such a commodity (this will be discussed in the following part of this section). More analysis is needed to determine how such parameters can be obtained by means of surveys or questioning public officials.

Up to this point only the consumption side of the economy has been discussed. The regional economy is further assumed to possess a finite number of production processes. In the real world a firm or company may possess one or more of these processes. One may interpret a production process as an available technology. Essentially, a production process is a method of transforming one or more commodities into one or more different commodities.

For this paper it is assumed that each production process can be specified either as an activity vector, a single output linear homogeneous Cobb-Douglas production function, or as a single output linear homogeneous C.E.S. production function. Each of these

specifications displays constant returns to scale, for example, if all inputs were doubled, output would also be doubled. The assumption of constant returns to scale can be shown to imply that at equilibrium each production process operated at a positive level will be associated with zero profits. Thus ownership of the means of production need not be specified since the problem of distribution of profits among consumers has been assumed away.

While the assumptions of constant returns to scale may appear to be a very strong assumption, in practical applications a process which does not display constant returns to scale could be approximated by a set of constant returns to scale production processes. Again the assumptions of perfect competition prevail on the input and product markets so that the operators of a production process are assumed to be price takers. If a price system p prevails on the market, the operator of each production process will calculate its profits at these prices. For a production process of the activity type profits are very easy to calculate. First the following convention is adopted: let a_i be the i^{th} element in an activity vector. If $a_i < 0$, commodity i is an input to the production process; if $a_i > 0$, commodity i is an output to the production process; if $a_i = 0$, commodity i is not involved in this production process. With this convention the profit associated with activity a at the price system p is $p \cdot a = \sum_{k=1}^{n} p_k a_k$.

Now when production is specified by a single output linear homogeneous Cobb-Douglas production function written:

$$P(x) = A \prod_k (x_k)^{b_k} \quad \text{with} \quad \sum_k b_k = 1$$

or by a single output linear homogeneous C.E.S. production function written:

$$P(x) = \left(\sum_k (b_k)^{1-a} (x_k)^a \right)^{1/a}$$

where $e = 1/(1-a)$ is the constant elasticity of substitution, there is a two step procedure involved for determining profits. First at the price system p, the optimal activity vector must be determined. Mathematically there are two methods by which one may determine the optimal input-output (activity) vector at unit output. The first method is to minimize costs $C = \sum_k p_k x_k$ subject to being on the unit isoquant $P(x) = 1$. The second method is to assume some arbitrary cost constraint C; maximize output subject to this cost constraint; determine the level of output so obtained; and then to adjust by a change of scale the input demands to produce unit output. Both of these methods are applied to these two production functions in Appendix A and they both give the following identical results; for the Cobb-Douglas production function the j^{th} element in the derived optimum activity vector can be determined by:

$$x_j = b_j \prod_k (p_k)^{b_k} / \; p_j A \prod_k (b_k)^{b_k}$$

provided $p_j \neq 0$ in which case x_j is a free input good which does not appear on the market, while for the C.E.S. production function

the j^{th} element will be given by:

$$x_j = b_j/(p_j)^e \left(\sum_k b_k (p_k)^{1-e} \right)^{1/a}$$

provided $p_j \neq 0$ in which case x_j is a free input good which does not appear on the market. Since these are input coefficients, they will be entered into the derived activity vector with a negative sign. Since these input demand functions have been determined for a unit level of output, the positive 1 is entered as the coefficient of the commodity being produced. Again if we represent the j^{th} element of such a derived vector by a_j, the profit associated with the derived optimal activity will be $p \cdot a = \sum_{k=1}^{n} p_k a_k$. Clearly at any price system p, profits can be calculated for every production process and one of these processes will maximize profits at such a price system (if more than one process maximizes profits, one process can be chosen arbitrarily). We will see that this will be necessary in the computation of equilibrium prices.

Note that these input demand functions are homogeneous of degree zero in the prices. As in the case of the consumer demand functions, only relative prices need be considered in determining these input demands. Hence, the only price systems that need be considered in the analysis are those whose components add up to unity.

The production sector will also contain a method of free disposal of any commodity. Specifically, if there are n commodities, there will be n disposal activities, one for each commodity. The disposal

activity for commodity j will be of the form $(0, \ldots, -1, \ldots, 0)$ with the -1 in the j^{th} position and zero in all other components.

We will also assume that production is irreversible, i.e., there is no production process or combination of production processes from which one can retrieve the inputs to production from the outputs of production. In a mathematical statement, the assumption of irreversible production is equivalent to saying that the set of production activities and derived production activities is positively linearly independent. This assumption guarantees that for any fixed amount of resources in the economy, the production set will be bounded.

The general equilibrium problem can now be interpreted in the following way. A price is quoted in the market place for every commodity involved in trade. Every economic agent is fully aware of each and every price. Each consumer sells all of his initial holdings at this price system to some central agency and obtains money in exchange. The producers buy some of these commodities as inputs to production and transform them into ouput commodities which consumers then purchase on the product market. The producers choose the level of operation at which their profits are maximized. Now it may happen that at this set of prices the consumers want to buy more of a particular good than is available, or some good will be in excess supply after all the purchases are made. If this happens the prices quoted in the market cannot be equilibrium prices. A new set of prices is called and the same process is repeated. A general equilibrium price system will then be that

vector of prices at which: (1) each consumer maximizes his preferences subject to his budget constraint, (2) each producer maximizes his profits, and (3) the market is cleared and consumers spend all their income, i.e., Walras' law holds. The assumption of free disposal guarantees that all the prices will be nonnegative. Indeed, if a free disposal activity is operated at a positive level at equilibrium, the commodity being disposed of is relatively abundant and will have a zero price since there is no demand for it and it can be disposed of at no cost.

The general equilibrium solution determines the optimum level of operation of each of the possible production processes. Those production processes which are inefficient will have an optimum level of operation of zero and should be shut down. As a specific example, there may be a number of different technologies available for cleaning up the pollution in a river. The general equilibirum solution would determine which technology or combination of technologies should be employed.

In some applications one may wish to have one of the consumers be the government. This has the advantage that one can incorporate the preference ordering of the central planners into the analysis. One may wish to weight more heavily the preferences of the government than its income alone permits. This may be accomplished by taxing the income of the other consumers and letting that income be spent by the government. A simple percentage income tax for each consumer has been programmed into the model. The details on how to use this will be found in Appendix B.

In concluding this section on the assumptions of general equilibrium analysis, the flexibility of this analysis should be noted. First, a time dimension (dynamics) can be introduced by dating the commodities; for example, bread at time i could be considered different from bread at time j. Such a technique permits the introduction of production technologies which require more than one time period to complete. It also permits the study of implicit interest rates and interest costs on commodities available for future consumption. Secondly, a spacial dimension can be added for selected variables; for example, water at location i can be considered a distinct commodity from water at location j. Finally, quality distinctions can be made in commodities by treating different quality ranges as separate variables. For example, water of a very good quality could be defined as a different commodity from water of a poor quality.

The Representation of a Public Good in General Equilibrium Analysis

Today there is much concern for the betterment of society. Many of the needs of society fall in a class of commodities which have come to be called public goods. A public good can be defined as any commodity which is consumed equally by all members of the society and of which each member consumes the total amount in existence [3, p. 49]. Examples of public goods are such items as national defense, clean air, sanitation services, etc. The basic problem is that different individuals generally prefer different amounts of this type of good. How much public goods should the government provide is always a difficult question.

From the point of view of general equilibrium analysis, a public good associated with a given consumer must be considered as distinct from the same public good associated with any other consumer. In production, the same number is then repeated as many times as there are consumers in the economy. In consumption, the consumer has a positive demand only for the public good associated with him, and a zero demand for the same public good associated with any other consumer. For example, if there are ten consumers and one public good in the economy, the commodity vectors must contain ten distinct components associated with the public good, one for each consumer. If a production activity produces one unit of a public good, this production activity must represent ten outputs, one for each consumer. The production processes used in the algorithm of the Fundamental Theorem can thus be specified directly as activity vectors or be derived from a one commodity continuous linear homogeneous production function of its inputs.

Any general equilibrium solution associates a relative price with each commodity in the economy. Clearly then, there is a price for each public good for each consumer. Since in general, different consumers have different demands for the same public good, the relative price of each public good will differ for each consumer. The price system is such that each consumer demands the entire amount of the public good associated with him. Indeed, general equilibrium price systems clear the market, i.e., supply equals demand.

The resultant prices associated with a public good are usually shadow prices, since in general consumers do not pay directly for them;

public goods are usually paid for by taxes. The prices of public goods reflect the desire or preference of each consumer for the public good and hence, can serve as a basis for a tax system based on consumer preferences.

In general, the mathematical representation of an economy with public goods is as follows. A similar representation has been developed independently by Duncan K. Foley [2]. There are n private goods, q public goods, m non-disposal production activities and r consumers in the economy.

Each consumer is endowed with an initial bundle of resources, that is, a finite amount of the private goods which he can buy and sell at the current prices on the market and from which he derives part of his income, and a finite amount of the public goods which can be bought and sold on the market and from which each consumer derives part of his income based on his own price of a public good. A typical initial bundle of resources for consumer t is an $rq + n$ dimensional column vector with zeros in $(r-1)q$ of the first components and nonnegative numbers in the it^{th} component for $i = 1, \ldots, q$ and in the last n components, for example $w^t = (0, \ldots, w_{1t}, \ldots, 0; \ldots; 0, \ldots, w_{qt}, \ldots, 0; w_{q+1,t}, \ldots, w_{q+n,t})$. Furthermore, each consumer has well-defined preferences for private goods and for public goods alike, represented by a utility function homogeneous of degree one in the commodities. Each consumer's preferences are assumed independent of the preferences of all other consumers and no consumer can consume the public good with another consumer's label. Thus each consumer's

utility and demand functions contain no other public good but that with his own label.

The productive sector of the economy is specified by a set of m activity vectors whose negative elements represent inputs and whose positive elements represent outputs. Each production activity which produces public goods repeats the positive component corresponding to a public good r times, one for each consumer. A typical production activity with public goods is an $rq + n$ column vector of the form $a^j = (a^j_{11}, \ldots, a^j_{1r}; \ldots; a^j_{q1}, \ldots, a^j_{qr}; a^j_{q+1}, \ldots, a^j_{q+n})$ where $a^j_{k1} = \ldots = a^j_{kr} > 0$ if the k^{th} public good is produced in the j^{th} production activity, $a^j_{k1} = \ldots = a^j_{kr} < 0$ if the k^{th} public good is used as an input in the j^{th} production activity, $a^j_{k1} = \ldots = a^j_{kr} = 0$ if the k^{th} public good is not involved in the j^{th} production activity, $a^j_{q+i} > 0$ represents an output of private good i, $a^j_{q+i} < 0$ represents an input of private good i, and $a^j_{q+i} = 0$ if private good i is not involved in the j^{th} production activity.

At equilibrium, there exists an $rq + n$ dimensional price vector $p = (p_{11}, \ldots, p_{1r}; \ldots; p_{q1}, \ldots, p_{qr}; p_{q+1}, \ldots, p_{q+n})$ such that for each production activity a^j operated at a positive level we have that:

$$\sum_{k=1}^{q} \sum_{t=1}^{r} p_{kt} a^j_{kt} + \sum_{i=1}^{n} p_{q+i} a^j_{q+i}$$

$$= \sum_{k=1}^{q} a^j_{kt} \sum_{t=1}^{r} p_{kt} + \sum_{i=1}^{n} p_{q+i} a^j_{q+i}$$

$$= \sum_{k=1}^{q} p_k a_k^j + \sum_{i=1}^{n} p_{q+i} a_{q+i}^j = 0$$

since $a_k^j = a_{k1}^j = \ldots = a_{kr}^j$ and where $\sum_{t=1}^{r} p_{kt} = p_k$ is the unit value of the k^{th} public good.

Now for each consumer t we have by Walras' law that:

$$\sum_{k=1}^{q} p_{kt} D_k^t(p) + \sum_{i=1}^{n} p_{q+i} D_{q+i}^t(p) = \sum_{k=1}^{q} p_{kt} w_{kt} + \sum_{i=1}^{n} p_{q+i} w_{q+i,t}$$

so that:

$$\sum_{i=1}^{n} p_{q+i} D_{q+i}^t(p) = \sum_{i=1}^{n} p_{q+i} w_{q+i,t} - \sum_{k=1}^{q} p_{kt} \left[D_k^t(p) - w_{kt} \right]$$

and the tax burden each consumer t is willing to assume is given by $\sum_{k=1}^{q} p_{kt} \left[D_k^t(p) - w_{kt} \right]$, the value of his excess demand for public goods beyond what he already enjoys. Hence p_{kt} can be interpreted as a discriminatory specific tax on each unit of the k^{th} public good for the t^{th} consumer. This result apparently contradicts Foley's conclusion [3, p. 57], that in the case of public goods there does not exist a zero net tax point in which each person's benefits from the public goods exactly equal his tax burden. However, Foley deals with an income tax structure while the tax structure in the present public goods' model is a specific discriminatory tax. This tax may be collected by a government which then gives this money to the firm which produces the k^{th} public good or else this tax may be levied directly by the producer of the k^{th} public good.

Now by Walras' law the society as a whole must use up all of its income so that adding the above equations over all the consumers we obtain:

$$\sum_{i=1}^{n} p_{q+i} \sum_{t=1}^{r} D_{q+i}^{t}(p) = \sum_{i=1}^{n} p_{q+i} \sum_{t=1}^{r} w_{q+i,t} - \sum_{t=1}^{r} \sum_{k=1}^{q} p_{kt} \left[D_{k}^{t}(p) - w_{kt} \right]$$

Let $\sum_{t=1}^{r} D_{q+i}^{t}(p) = D_{q+i}(p)$ be the market demand for private good i, $\sum_{t=1}^{r} w_{q+i,t} = w_{q+i}$ be the initial amount of commodity i in the regional economy, and since at equilibrium the supply of public goods must equal its demand and the total amount of the public good is consumed by each consumer, we obtain:

$$\sum_{i=1}^{n} p_{q+i} D_{q+i}(p) = \sum_{i=1}^{n} p_{q+i} w_{q+i} - \sum_{k=1}^{q} p_{k} \left[D_{k}(p) - w_{k} \right]$$

where $\sum_{t=1}^{r} p_{kt} = p_{k}$ and $w_{k} = \sum_{t=1}^{r} w_{kt}$. This states that $\sum_{k=1}^{q} p_{k} \left[D_{k}(p) - w_{k} \right]$ is the total tax burden the society is willing to assume for the optimum quantity of the k^{th} public good. This implies that p_{k} is the specific tax for the k^{th} public good for society as a whole and then $(p_{11}, \ldots, p_{1r}; \ldots; p_{k1}, \ldots, p_{kr}; \ldots; p_{q1}, \ldots, p_{qr})$ is the specific tax structure in which each consumer's benefits exactly equal his tax burden.

Allocation of Money Resources

To a Regional Economy

The use of general equilibrium analysis in applied economics is limited by the fact that most economies are so large that the specification requirements are prohibitive. Isolating a small region for analysis is restricted by the fact that there is no guarantee that the region will ever be in equilibrium with the economy in which it is embedded.

This subsection proposes a method of utilizing the powerful tools of general equilibrium analysis in studying a small regional economy whose boundaries are determined solely on the basis of the objective of the analysis and on the feasibility of the required specifications. It opens up the possibility of applying general equilibrium analysis to economic areas such as cities, small geographic areas, and so on. Specifically the Fundamental Theorem is applied to an interaction mechanism which brings the small, artificially bounded economy into equilibrium with the economy in which it is embedded. This is an entirely new application of the Fundamental Theorem.

In order to simplify the nomenclature, the small economy identified for analysis will hereafter be called the "regional" economy and the economy in which this small economy is embedded will be called the "larger" economy. Since by assumption the regional economy is embedded in the larger economy, the regional economy is in a non-optimum state if the relative prices of the commodities which are in both the

regional and the larger economies differ in these two economies. Indeed, if the relative prices do not match then it is feasible for some entrepreneur to improve his own position without hurting anyone else simply by buying those commodities which are relatively inexpensive in one economy and selling them in the other economy. Clearly if the relative prices in the regional economy are exactly matched with the corresponding relative prices in the larger economy, this type of arbitrage is not feasible and the regional economy is in a Pareto optimal state relative to the larger economy. The exact matching of these relative prices will be the goal of the interaction mechanism.

In most cases, it will be possible to exactly match the relative prices in the regional economy with the relative prices of the corresponding commodities in the larger economy. Whenever the interaction mechanism does not bring the regional economy into full equilibrium with the larger economy, it still produces the highest level of utility attainable for each consumer in the regional economy consistent with general equilibrium in the regional economy itself, i.e., a Pareto optimal state consistent with the structure of the economy. This will be discussed in detail below.

To summarize, we say that the regional economy is optimized with respect to the larger economy if the relative prices in the regional economy are as close as possible to the relative prices of the corresponding commodities in the larger economy.

The novelty of this optimization procedure arises from the fact that the regional economy is treated as an integral part of the larger economy. For this reason, the concept of optimization relates to equating relative prices in the two economies. Were the two economies considered as distinct, such as the economies of two countries, then political and trade barriers would in general preclude the matching of relative prices. In this case, there is a true cost of trading between the two countries and the adjustment mechanism proposed in this subsection no longer applies.

We will assume that the regional economy can be completely described by a set of conditions which are sufficient for the application of the algorithm to compute approximate equilibrium prices and quantities. Specifically assume that the regional economy consists of a finite number of commodities and of a finite set of producers and consumers. Each producer is completely described by a finite set of productive activities which display constant returns to scale. These can be represented either by activity vectors or by single output production functions linear homogeneous in all inputs. Each consumer is initially endowed either with a limited amount of physical resources or money or both. Moreover, each consumer possesses a well-defined utility function linear homogeneous in all commodities, from which he derives his demand for each commodity as a function of the prices in the regional economy.

Trade with the larger economy is introduced in the following way. At least one consumer initially holds a positive amount of money. This

assumption could be relaxed by requiring one or more of the consumers to obtain an initial quantity of money by selling some of their initial resources to the larger economy. The money can be used only to increase the initial quantity of physical resources by purchases from the larger economy at existing prices there. Physical resources initially held by consumers can also be sold at existing prices in the larger economy. This transaction will result in an increase in the amount of money initially held by the consumers; however, the consumer must immediately spend this additional money on physical resources in the larger economy. In other words, all monetary transactions are completed before equilibrium prices and quantities are computed in the regional economy. Thus, each consumer's wealth constraint depends only on the quantity of physical resources he finally holds after all monetary transactions have been completed.

The rule for allocating the money resource to purchases and sales of commodities in the larger economy is defined as follows. There exists a central decision-maker who dictates how each and every consumer in the regional economy is to allocate his money holdings in trades with the larger economy. Hereafter this central decision-maker, who may or may not be one of the consumers in the regional economy, will be referred to as the government. The dictates of the government will be represented by a set of real numbers, one for each commodity, which add up to one. This set of numbers is then called an allocation of initial money resources.

The numbers which define an allocation of initial money resources can be negative, positive, or zero so long as they add up to one. A negative number a_i represents an instruction from the government to the consumer to sell in the large economy the number of units of commodity i which corresponds to a_i multiplied by the amount of money he holds initially, at the price of commodity i in the larger economy. This quantity of commodity i is now subtracted from the initial amount of commodity i the consumer initially holds. If, however, the consumer is instructed to sell more units than he holds initially, the excess value creates an exogenous demand for commodity i in the regional economy at the price of commodity i in the regional economy. This exogenous demand can be interpreted as a demand coming from the larger economy. This transaction is performed first and the proceeds from the sale increase the amount of money with which the consumer can buy other commodities in the larger economy. A positive number a_j represents an instruction from the government to the consumer to buy in the larger economy the number of units of commodity j which correspond to a_j multiplied by the amount of money he holds initially at the price of commodity j in the larger economy. This additional quantity of commodity j is now added on to the initial amount of commodity j the consumer initially holds. A zero number a_k is an instruction from the government to the consumer to keep the initial quantity of commodity k unchanged.

For example suppose there are three commodities in the regional economy with corresponding per unit prices $p_1 = \$1.00$, $p_2 = \$2.00$, $p_3 = \$3.00$,

respectively, in the larger economy, and the consumer owns \$100 in money resources. Let the allocation of money resources dictated by the government be given by the numbers $a_1 = 0.4$, $a_2 = -0.3$, and $a_3 = 0.9$. Then the consumer is instructed to sell $(a_2)(\$100) = (-0.3)(\$100) = \$30$ worth, i.e., 15 units of commodity 2, in the larger economy. He now has \$130 in money resources with which he will buy $(a_1)(\$100) = (0.4)(\$100) = \$40$ worth, i.e., 40 units of commodity 1 and $(a_3)(\$100) = (0.9)(\$100) = \$90$ worth, i.e., 30 units, of commodity 3 from the larger economy. The consumer now has 40 more units of commodity 1, 15 less units of commodity 2, and 30 more units of commodity 3 in his initial vector of physical resources. If, however, he owned initially less than 15 units of commodity 2, say only 10 units, he now has 0 units of commodity 2 in his initial vector of physical resources, and there has been created in the larger economy a demand for 5 units of commodity 2 from the regional economy at the price of commodity 2 in the regional economy.

The underlying assumption is that the regional economy is so small with respect to the larger economy that the larger economy can supply any amount of any commodity which consumers in the regional economy may wish to buy from it. Thus the minimum size of the larger economy is indirectly specified by the magnitude of the numbers in the allocation of money resources and by the amount of money initially held by all consumers in the regional economy.

A negative number in the allocation of money resources and the amount of money initially held by all the consumers in the regional economy

determine the quantity of the corresponding commodity that must be sold in the larger economy. Thus, given the amount of money initially held by all the consumers in the regional economy, the admissible lower bound of any negative number in the set of all allocations of money resources specifies the maximum quantity of the corresponding commodity which the larger economy can buy from the regional economy. The extent to which the larger economy can buy from the regional economy will be called the absorptive capacity of the larger economy. Clearly, the choice of admissible allocations of money resources in the regional economy will depend on the absorptive capacity of the larger economy.

To each allocation of money resources there corresponds a uniquely well specified regional economy. Indeed, after the initial transactions have taken place with the larger economy, the consumers in the regional economy own a new set of physical resources and have demand functions homogeneous of degree zero in all prices; the exogenous demand which may have been created by trade with the larger economy is simply added onto the total demand for each commodity; the set of production processes display constant returns to scale as required. Thus, the conditions for applying the algorithm to compute a set of equilibrium prices and quantities are all verified. Consequently, to each allocation of money resources, there will correspond a unique set of equilibrium prices.

We will now say that the regional economy is optimized with respect to the larger economy if the relative prices of some or all the commodities

are as close as possible to the relative prices of the same commodities in the larger economy. The rationale behind this concept of optimality is that the corresponding general equilibrium will then minimize the dislocative effects upon the larger economy of any development taking place in the regional economy. It is a constrained optimum subject to the absorptive capacity in the larger economy.

Not every commodity in the regional economy need be a commodity in the larger economy. Not every commodity in the regional economy need be marketable in the larger economy. For example, water in a river basin is localized in that particular region through which the river flows and is usually marketable there only. Such commodities will have no corresponding number in the allocation vector since there is no price in the larger economy to compare its price with. Thus, the set of relative prices in the regional economy which will be compared with the corresponding relative prices in the larger economy will only be a subset of the set of equilibrium prices computed by applying the general equilibrium algorithm to the regional economy. This presents no difficulty, however, since relative prices can always be scaled differently without changing their relationship to one another. The algorithm which will be devised in the next section to find the appropriate allocation of money resources will simply require that the sets of prices to be compared be scaled to add up to unity in both the regional economy and the larger economy.

Heuristically, there are two basically distinct situations arising from the analysis of the possible results of this model of money

resource allocation by government fiat. In the first instance, the only commodities that are common to both the regional and the larger economy are inputs to production in the regional economy. If there is sufficient absorptive capacity for these commodities in the larger economy, then an appropriate allocation of money resources to the factors of production will result in matching exactly their relative prices in both economies.

Indeed, the price of any factor of production can be made as high as desired in the regional economy by making that particular commodity sufficiently scarce. Any negative allocation of money resources to a particular commodity is a mandate of the government to sell a corresponding quantity of the commodity. If this quantity exceeds the initial stock held by any consumer with money wealth, an exogenous demand is created to the amount by which the quantity to be sold exceeds the initial stock. Clearly, larger negative allocations produce a greater scarcity and a higher exogenous demand which, together with the endogenous demand, result in higher prices. Thus if an appropriate quantity of any factor of production is sold in the larger economy, its price can be made arbitrarily high. The only restriction is the absorptive capacity of the larger economy.

The price of any factor of production can be made as low as desired in the regional economy by making that particular commodity sufficiently abundant. Any positive allocation of money resources to a particular commodity is a mandate of the government to buy a corresponding quantity of the commodity from the larger economy at the

fixed price prevailing in that economy. Clearly, larger positive allocations produce a greater abundance which results in lower prices. If there are sufficient money resources in the regional economy, enough of any particular commodity can be purchased in the larger economy to make its relative price in the regional economy as low as desired.

A completely different situation arises when there are one or more commodities common to both the regional and the larger economy that are outputs to production in the regional economy. In this case, it is not always possible to match exactly the relative prices in both economies. Consider, for instance, a regional economy with a productive activity all of whose components (inputs and outputs) are common to both the regional economy and the larger economy, and which is profitable at the prices prevailing in the larger economy. It will then always be better for the regional economy to operate the profitable activity at a higher level of operation, buying more inputs and selling more outputs in the larger economy. This is a consequence of the assumption of constant returns to scale in production which can then be bounded only by a fixed supply of physical input resources. By allowing negative allocations of the money resources thereby increasing the positive allocations, there is no bound on the quantity of physical input resources that the regional economy can acquire, except the absorptive capacity of the larger economy. Thus, each profitable productive activity will be scaled up to the bounds defining the absorptive capacity of the larger

economy. The relative prices in the regional economy will be as close as possible to the relative prices in the larger economy subject to the absorptive capacity of the larger economy, but they will not be matched exactly.

The opposite situation arises when all the productive activities in the regional economy are composed of commodities common to both economies but no productive activity is profitable at the relative prices prevailing in the larger economy. This says that production in the regional economy is completely inefficient. The consumers in the regional economy would be better off to sell their factor resources and buy the outputs at the given fixed prices in the larger economy. If there is enough absorptive capacity in the larger economy production in the regional economy will grind to a halt. With sufficient absorptive capacity in the larger economy, relative prices in both economies will be exactly matched.

A more complex but perhaps more realistic situation develops when some but not all of the outputs in the production acitivites are common to both the regional economy and the larger economy. Since at least one price is left to vary independently, an activity which is profitable at one allocation of the money resources is not necessarily profitable at some other allocation. Specifically, consider an activity which produces joint outputs such that one output is localized within the regional economy. If excluding the localized commodity, the activity is profitable at the relative prices prevailing in the larger economy, the situation reduces to

the case previously considered and the results depend entirely on the absorptive capacity of the larger economy. If not, because of constant returns to scale, as the level of operation of any such activity is expanded, the supply of the localized output increases and its prices falls within the regional economy. Equilibrium is attained when the price of the localized commodity is reduced to the point that the productive activity yields exactly zero profits in the regional economy and the relative prices in the regional economy of the commodities common to the larger economy are identical to the relative prices of the same commodities in the larger economy.

A similar situation arises when one of the factors of production is a physical resource localized to the regional economy. In this case, as production expands by purchasing other factor inputs and selling outputs at a profit in the larger economy if need be, the localized factor input becomes relatively more and more scarce so that its price increases until all corresponding productive activities are no longer profitable at the relative prices in the regional economy. Again, provided there is sufficient absorptive capacity in the larger economy, equilibrium will be attained so that the relative prices of the commodities common to both economies are matched exactly and the localized factor price is increased to the point that the most profitable activities all yield zero profits.

The problem for the government is now clear. It must determine the optimum allocation of the money resource so that wherever possible the prices of those commodities common to both the regional and

larger economies will be identical in both places. The algorithm presented in the next section will determine for the government this optimum allocation.

A Numerical Example

To illustrate the applicability of the allocation model to an economy with a public good, consider a small hypothetical economy which displays alternatives similar to those present in the real world. The specifications of the economy are presented in Table 1. The economy is assumed to have four factors of production: unskilled labor, technical labor, steel, and cement; an intermediate good labeled capital; a consumer good which is locally produced; and a public good, water, which can be consumed by each consumer without affecting the quantity or quality available to any other consumer. This public good is therefore indexed separately as three distinct commodities, one index for each consumer. Units of water are interpreted either as quality units or as quantity units. If, for instance, the water is assumed to be used in a river with a relatively fixed flow, or in a lake, then the units of measurement can be quality standards (such as units of dissolved oxygen or some other method of classifying water purity); if the water is simply drawn from a lake or from a river, then the units can be measures of quantity. No attempt is made to formulate a real economic problem so that the reader may supply his own economic interpretation.

TABLE 1

SPECIFICATION OF A HYPOTHETICAL REGIONAL ECONOMY WITH A PUBLIC GOOD

Economic Agent	Function Type	Un-skilled Labor	Tech-nical Labor	Steel	Cement	Capital	Local Good	Water C1	Water C2	Water C3	Elasti-city or Scale
Production Functions for Each Firm in the Economy											
Capital Good Production	C.E.S.	0.3	0.5	0.7	0.2	-1.0	0.0	0.0	0.0	0.0	0.3
Local - High Pollution	Activity	-1.2	-0.5	0.0	0.0	-1.0	4.0	-2.0	-2.0	-2.0	
Local - Moder-ate Pollution	Activity	-1.4	-1.3	0.0	0.0	-2.0	4.0	-0.5	-0.5	-0.5	
Local - No Pollution	Activity	-0.5	-0.5	0.0	0.0	-4.0	4.0	0.0	0.0	0.0	
Dam	Activity	-0.8	-0.8	-0.9	-1.0	0.0	0.0	1.0	1.0	1.0	
Pond	Activity	-2.3	0.0	-0.9	-1.0	0.0	0.0	1.0	1.0	1.0	
Chemical	Cobb-Douglas	0.1	0.5	0.2	0.2	0.0	0.0	-1.0	-1.0	-1.0	1.0
Resources Available to Each Consumer in the Economy											
Consumer 1		80.0	0.0	0.0	0.0	0.0	3.0	4.0	0.0	0.0	
Consumer 2		0.0	40.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	
Consumer 3		0.0	0.0	50.0	60.0	1.0	2.0	0.0	0.0	4.0	
Utility Functions for Each Consumer in the Economy											
Consumer 1	Cobb-Douglas	0.0	0.0	0.0	0.0	0.0	0.6	0.4	0.0	0.0	1.0
Consumer 2	C.E.S.	0.0	0.0	0.0	0.0	0.0	4.0	0.0	6.0	0.0	3.0
Consumer 3	C.E.S.	0.0	0.0	0.0	0.0	1.0	4.0	0.0	0.0	6.0	0.8

Money Resources Available to Each Consumer

Consumer 1	Consumer 2	Consumer 3
$100	$200	$300

The production sector contains alternative methods of production. There is only one production process which manufactures the capital necessary for producing the local consumer good. This process allows some input substitution and is represented by a single output linear homogeneous C.E.S. production function with an elasticity of substitution of 0.3. Three production processes exist for the production of the local consumer good, each represented by a fixed activity vector. The first process can be considered a high pollution process since it uses up 2 units of water for each unit level of operation. It is, however, relatively inexpensive in terms of its other inputs since it requires only one unit of capital. The second process produces a moderate level of pollution using only 0.5 units of water at unit level of operation. This process is more expensive, for besides requiring an additional unit of capital, it also has a higher technical labor input requirement. The third process is the most expensive in terms of capital investment. However, it produces no pollution at all.

Now the public good, water, is demanded by all three consumers and there exist three different methods of producing this public good. The first method, identified as a dam, requires approximately an even amount of the four factors of production to produce one unit of water. The pond method is a highly non-technical method requiring relatively high levels of unskilled labor. The third alternative is a chemical process which is technically oriented and exhibits a unitary elasticity of input substitution. This process is represented mathematically as a Cobb-Douglas production function.

Each consumer is assumed to have a fixed supply of physical resources as presented in Table 1. Note that each consumer is initially endowed with the same number of units of the public good. These are represented as the 4.0 of Water C1 available to Consumer 1, 4.0 of Water C2 available to Consumer 2, and the 4.0 of Water C3 available to Consumer 3.

The demand sector is represented by utility functions, one for each consumer, of either the Cobb-Douglas or C.E.S. type. Note again that by definition of a public good each consumer demands only his own public good. Each consumer has a demand for the public good and the local good; the third consumer also has a demand for capital.

This small economy is assumed to be embedded in some larger economy with fixed dollar prices for those commodities which can be traded with the regional economy. Each consumer in the regional economy possesses some dollars which he can use in trade with the larger economy. It is further assumed that a government exists which issues a mandate to each consumer as to how he should allocate his money resource. The objective of this mandate is to equate the relative prices in the regional and in the larger economies of the commodities which can be traded between the two economies.

The allocation model is applied to this problem under two separate assumptions as to which commodities can be traded between the two economies. Table 2 presents the case where the four factors of production can be traded while in Table 3 it is assumed that unskilled labor cannot be imported or exported while the local consumer good can.

TABLE 2a

RESULTS OF ALLOCATION MODEL APPLIED TO THE ECONOMY OF TABLE 1 UNDER THE ASSUMPTION THAT ONLY THE FACTORS OF PRODUCTION CAN BE TRADED

Allocation Grid = 50, General Equilibrium Grid = 100

	Unskilled Labor	Technical Labor	Steel	Cement
Absolute Price	$3.00	3.00	3.00	3.00
Relative Price	.250	.250	.250	.250
True Boundaries	-20	-20	-20	-20
Computational Boundaries	-5	37	10	0
Equilibrium Allocation	-.049	.785	.223	.040
Equilibrium Prices	.252	.252	.239	.258

Final General Equilibrium Solution

Firms in Final Equilibrium Solution	Levels of Operation
Capital Good Production	46.5
Local - Moderate Pollution	18.9
Chemical	82.5

Commodities	Gen. Equil. Computational Boundaries	Equilibrium Relative Prices	Estimated Final Demand	Estimated Net Supply
Unskilled Labor	0	.087	8.1	6.7
Technical Labor	0	.087	0.0	4.4
Steel	0	.082	0.0	-5.4
Cement	0	.089	0.0	2.3
Capital	0	.183	10.1	9.8
Local Good	0	.181	86.9	84.6
Water-Consumer 1	0	.053	79.1	77.1
Water-Consumer 2	0	.109	78.6	77.1
Water-Consumer 3	0	.129	80.0	77.1

Note 1: The barycenter averaging technique was used for averaging the equilibrium price simplices.

Note 2: The Levels of Operation are the weights associated with the respective activity vectors in the final nonnegative solution to $Bx=w$. The Estimated Net Supply was determined by multiplying the levels of operation by the activity vectors calculated at the Equilibrium Relative Prices.

TABLE 2b

RESULTS OF ALLOCATION MODEL APPLIED TO THE ECONOMY
OF TABLE 1 UNDER THE ASSUMPTION THAT ONLY
THE FACTORS OF PRODUCTION CAN BE TRADED

Allocation Grid = 100, General Equilibrium Grid = 100

	Unskilled Labor	Technical Labor	Steel	Cement
Absolute Price	$5.00	3.00	1.00	6.00
Relative Price	.333	.200	.067	.400
True Boundaries	-40	-40	-40	-40
Computational Boundaries	-26	85	40	-20
Equilibrium Allocation	-.168	.875	.411	-.118
Equilibrium Prices	.335	.201	.066	.398

Final General Equilibrium Solution

Firms in Final Equilibrium Solution	Levels of Operation
Capital Good Production	103.9
Local - No Pollution	23.2
Chemical	85.9

Commodities	Gen. Equil. Computational Boundaries	Equilibrium Relative Prices	Estimated Final Demand	Estimated Net Supply
Unskilled Labor	11	.136	16.8	15.7
Technical Labor	4	.081	0.0	-.4
Steel	0	.027	0.0	2.0
Cement	13	.161	5.9	5.2
Capital	13	.153	12.9	12.1
Local Good	15	.171	109.1	101.8
Water-Consumer 1	3	.062	93.9	89.9
Water-Consumer 2	6	.091	103.5	89.9
Water-Consumer 3	8	.118	95.9	89.9

Note 1: The barycenter averaging technique was used for averaging the equilibrium price simplices.

Note 2: The Levels of Operation are the weights associated with the respective activity vectors in the final nonnegative solution to $Bx=w$. The Estimated Net Supply was determined by multiplying the levels of operation by the activity vectors calculated at the Equilibrium Relative Prices.

TABLE 3

RESULTS OF ALLOCATION MODEL APPLIED TO THE ECONOMY OF TABLE 1 UNDER THE ASSUMPTION THAT UNSKILLED LABOR IS LOCALIZED TO THE REGIONAL ECONOMY AND THE LOCALLY PRODUCED GOOD CAN BE TRADED

Allocation Grid = 100, General Equilibrium Grid = 100

	Technical Labor	Steel	Cement	Local Good
Absolute Price	$2.00	2.00	2.00	4.50
Relative Price	.190	.190	.190	.430
True Boundaries	-40	-40	-40	-40
Computational Boundaries	66	23	7	-15
Equilibrium Allocation	.689	.246	.085	-.020
Equilibrium Prices	.195	.184	.197	.423

Final General Equilibrium Solution

Firms in Final Equilibrium Solution	Levels of Operation
Capital Good Production	58.2
Local - Moderate Pollution	23.9
Chemical	102.0

Commodities	Gen. Equil. Computational Boundaries	Equilibrium Relative Prices	Estimated Final Demand	Estimated Net Supply
Unskilled Labor	9	.106	0.0	-1.7
Technical Labor	5	.083	0.0	3.4
Steel	6	.079	0.0	-2.8
Cement	6	.084	0.0	1.6
Capital	16	.181	11.2	11.5
Local Good	16	.181	104.0	101.8
Water-Consumer 1	3	.057	94.1	94.1
Water-Consumer 2	8	.106	98.2	94.1
Water-Consumer 3	10	.123	95.6	94.1

Note 1: The barycenter averaging technique was used for averaging the equilibrium price simplices.

Note 2: The Levels of Operation are the weights associated with the respective activity vectors in the final nonnegative solution to $Bx = w$. The Estimated Net Supply was determined by multiplying the levels of operation by the activity vectors calculated at the Equilibrium Relative Prices.

In all cases it is assumed that capital cannot be purchased or sold in the larger economy. The public good, water, is also treated as localized to the regional economy.

As pointed out in the last subsection, with sufficient absorptive capacity in the larger economy it will always be possible to match the relative prices of the factors of production if they are the only commodities traded between the two economies. Two different experiments are conducted under these specifications assuming different factor prices in the larger economy. In the first experiment each factor price is assumed to be \$3.00. Table 2a presents the results of this calculation. The equilibrium allocation indicates that unskilled labor should be exported, while technical labor, steel, and cement are to be imported into the region. With equal factor prices of \$3.00 the regional economy should produce 18.9 units of the locally produced good using the technique of moderate pollution. The chemical process should be used at the 82.5 level to produce the public good. At the final equilibrium prices the derived unit output activity for this chemical process is (-0.337, -1.685, -0.711, -0.657, 0.0, 0.0, 1.0, 1.0, 1.0). Similarly the unit output activity vector is calculated for the production of the capital good. At equilibrium this vector is (-0.375, -0.624, -0.888, -0.248, 1.0, 0.0, 0.0, 0.0, 0.0) and it is operated at level 46.5. Notice that the relative prices of the public good are substantially different for the different consumers. For the first consumer the relative price is only 0.053 while for the second consumer it is more than double that amount at 0.109, and for the third consumer

it is even higher at 0.129. At these different relative prices however, each consumer demands approximately the same number of units, i.e., approximately the net supply of 77.1.

Table 2b presents the same economic problem with different factor prices in the larger economy. A somewhat better matching of relative prices is achieved in this case using an allocation grid size twice as large as that used in the calculations of Table 2a. At this set of factor prices the regional economy should produce the local consumer good with the no pollution technology. The reason for this is that capital has become relatively less expensive, falling from 0.183 to 0.153. At this lower price of capital, the more capital intensive no pollution process becomes the most efficient. The larger net supply of water and of the local consumer good is due to the fact that the local economy is now wealthier with the value of its initial physical resources measured at the new prices prevailing in the larger economy. In this experiment, the local economy is optimized by exporting unskilled labor and cement and importing technical labor and steel.

The allocation model is also applied to this economy under a different assumption as to which commodities can be traded between the two economies. Specifically it is assumed that all the factors of production except unskilled labor, and the locally produced consumer good can be traded. Under this specification and with the prices prevailing in the larger economy as presented in Table 3, the allocation model indicates that the three factors of production should be imported and the locally produced consumer good should be exported. Production of

the local consumer good with the moderate pollution technology is carried out to the point where unskilled labor becomes relatively scarce and its price rises until the moderate pollution activity has exactly zero profits. A small part of this output is then sold in the larger economy.

Considerable computational efficiency is achieved by choosing computational boundaries for the general equilibrium solutions. Once it is known approximately where the allocation solution is to be found on the unit allocation hyperplane, small adjustments in the vector of initial physical resources do not greatly affect the equilibrium primitive set of prices provided there is enough continuity of the mapping. Therefore, at each iteration of the allocation model the entire price simplex need not be searched for the fixed point, but only that region in which the solution is expected to be found. This technique has the disadvantage that at some iteration of the allocation model an equilibrium primitive set of prices may contain a computational boundary element. If such an equilibrium price system is associated with an allocation in the final primitive set of allocations, the entire solution must be recomputed lowering sufficiently the general equilibrium computational boundaries so affected.

This section has applied a technique for determining the optimum production methods in a regional economy confronted with the problem that at least some of its methods of production "pollute" in the sense that they use up a public good. The quality of water or air, or the amount of other types of public goods produced depends on the interaction

between the demand for the public good and the demand for the product which uses up the public good in production. Which production technology to employ depends to a large extent on the techniques available to clean up the polluted public good.

One source of money for a region is the government. The government can be treated as an additional consumer with money wealth and a demand function reflecting its desire for cleaner water. Given the amount of money allocated to this region, the allocation model will then determine the optimum distribution of this money resource to the various commodities in the region. Hopefully, this methodology will make it possible to use the powerful tools of general equilibrium analysis in the search for the optimal allocation of money resources to regional developmental projects.

SECTION V

THE ALGORITHM OF THE FUNDAMENTAL THEOREM

This section will acquaint the reader with the algorithm which has been used both to compute equilibrium prices and to solve the allocation model. This section is not completely original as it is based on the work of Hansen and Scarf [5]. The originality consists in demonstrating that the vectors in P_k as defined for the Fundamental Theorem in computationally convenient form need not be taken only from the unit simplex as Hansen and Scarf have done; rather P_k can be vectors on an appropriately bounded subset of any hyperplane which is parallel to the unit simplex, and the conditions of the Fundamental Theorem will still hold. This extension is needed in the application of the algorithm to the allocation model.

There will be four parts to this presentation. The first part will contain a statement and proof of what will be called the Fundamental Theorem assuming degeneracy, an assumption which leads to a computationally convenient form of the algorithm. The second part will apply the Fundamental Theorem to the computation of general equilibrium prices. The third part will apply the Fundamental Theorem to the computation of an optimal allocation of money resources in a regional economy with an almost equivalent solution presented in part four.

The Fundamental Theorem

The first concept which must be understood is that of a primitive set. Let the vectors of a set P all be chosen from the G-hyperplane in

Preceding page blank

n-dimensional Euclidean space and be all vectors of the form $(k_1, k_2, \ldots, k_n)$ where $\sum_{i=1}^{n} k_i = G$, where G is any fixed integer and each k_i is an integer bounded from below by some integer Q_i unrestricted in sign. Clearly the G-hyperplane is the translate of a maximal proper linear subspace parallel to the unit hyperplane. Figure 1 gives the geometrical representation of a set P for the case $n = 3$, $G = 5$, $Q_1 = 0$, $Q_2 = 1$, and $Q_3 = -1$. Only the points at the vertices of the smaller triangles belong to the set P. Note carefully that there are many vectors in P with identical i^{th} coordinates. Geometrically all the points on a line parallel to any side of the larger triangle have one identical coordinate. Thus P so defined does not satisfy the assumption of nondegeneracy made in [9], and the replacement operation described there cannot be applied directly. There must be a method of comparing identical elements in a particular coordinate to determine their order. If an ordering can be devised which is a complete ordering, then primitive sets can be defined from this algebraic formulation of P even if the nondegeneracy assumption is violated. Specifically, Scarf and Hansen have chosen the following tie-breaking rule:

Definition: Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$. We define x_i to be larger than y_i (written $x_i \overset{i}{>} y_i$), if and only if, the vector $(x_i, \ldots, x_n, x_1, \ldots, x_{i-1})$ is lexicographically larger than $(y_i, \ldots, y_n, y_1, \ldots, y_{i-1})$. We also define y_i to be smaller than x_i (written $y_i \overset{i}{<} x_i$), if and only if, the vector $(y_i, \ldots, y_n, y_1, \ldots, y_{i-1})$ is lexicographically smaller than $(x_i, \ldots, x_n, x_1, \ldots, x_{i-1})$.

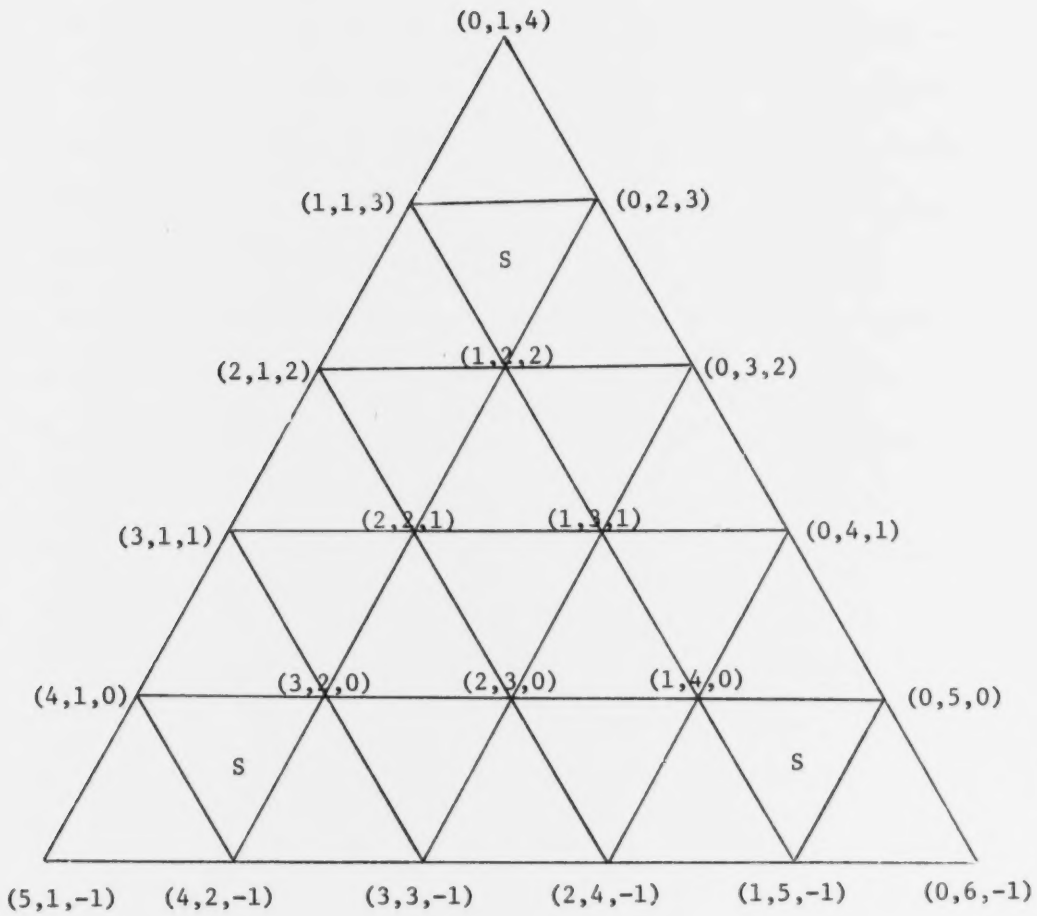

Fig. 1.--The set P formulated algebraically for the case of n=3, G=5, Q_1=0, Q_2=1, and Q_3=-1.

We may now define a primitive set: a subset Y of n distinct column vectors from P is said to be primitive if every column vector contains exactly one row maximizer according to the ordering $\geq_i$, and there is no vector belonging to the set P which is smaller (with respect to the ordering $\geq_i$) in all components than the row maximizer (with respect to the ordering $\geq_i$) of the set Y. For example, the vertices of any little triangle of Figure 1 provide a geometric example of a set of points from a set P taken from the G-hyperplane which determine a primitive set. To be consistent with the geometric objects called primitive sets in the case of nondegeneracy, only those little triangles whose sides are parallel to the sides of the large triangle are primitive sets. For those little triangles whose sides are not parallel to the sides of the larger triangle, it is the smallest triangle surrounding it whose sides are parallel to the sides of the large triangle, which forms the geometric primitive set. For example, the little triangle with vertices (2,2,1), (2,3,0), and (1,3,1) will not in itself be a primitive set. However, these vertices determine the primitive set whose vertices are (1,2,2), (3,2,0), and (1,4,0). Since we will be dealing only with the vertices which determine primitive sets, i.e., the vertices of the little triangles, we will hereafter refer to the little triangles as primitive sets.

In defining a primitive set Y above, it was specifically required that each column of Y contain exactly one row maximizer (according to ordering $\geq_i$). This defining property is needed to guarantee that

each primitive set in P will be determined by a unique subset of n elements from P. For example, without this property, the three points in Figure 1, (3,2,0), (2,2,1), and (1,3,1) would determine the same primitive set as the three points (2,3,0), (2,2,1), and (1,3,1), i.e., the primitive set with vertices (1,2,2), (3,2,0), and (1,4,0). This can be readily seen by drawing lines parallel to the sides of the large triangle through the three points in each set. The reader can easily verify that there are other three-point sets determining the same primitive set if the definition does not require specifically that each column of a primitive set Y contains exactly one row maximizer (according to the ordering $\geq_i$).

Anticipating the results of the following lemmata, it will be shown that the replacement operation, replacing a vector in a primitive set with another vector from the set P to form a new primitive set, is equivalent in geometric terms to lifting up one of the vertices of a little triangle in Figure 1 (the vector to be removed from the primitive set) and pivoting the entire triangle over the two remaining points, until the vertex which was lifted lies on top of the opposite vertex of the adjacent triangle. This vertex will then be the unique replacement vector. Specifically consider in Figure 1 the primitive set (3,2,0), (2,2,1), and (2,3,0) and the operation of removing the point (3,2,0) from this set. The unique replacement from P will be the vector (1,3,1).

While the theory of primitive sets with the unique replacement operation will be developed with the elements of this set P on the G-hyperplane,

it should be noted at the outset that in most applications known so far, and in particular for the applications of this paper, the set P_k actually used in the Fundamental Theorem will consist of vectors taken from the unit hyperplane. In fact, the theory developed for the set P will be shown to apply even more generally to a set P_k whose elements are taken from some H-hyperplane, where H is any rational number, i.e., if $p \in P_k$, $\sum_{i=1}^{n} p_i = H$. This will be accomplished by letting $\lambda = H/G$ and showing that the isomorphism which maps $(k_1, \ldots, k_n) \in P$ into $(\lambda k_1, \ldots, \lambda k_n) \in P_k$ will preserve primitive sets. With this mapping the elements of P_k will become closer to each other with larger values of G so that G will be called the grid size for the H-hyperplane.

We will now state and prove a series of lemmata which will characterize primitive sets in this algebraic formulation of P and which will demonstrate the technique of finding the unique replacement vector from P when a vector is removed from a primitive set. After demonstrating that this theory applies more generally to a set P_k derived from the set P by the isomorphism introduced above, the Fundamental Theorem will be proved using this algebraic formulation of P_k. The proofs are adapted from Hansen and Scarf [5].

To simplify the presentation, define the matrix K to be an $n \times n$ matrix with integral entries such that $k_{ij} \geq Q_i$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$ and whose column sums are identical, i.e., $\sum_{i=1}^{n} k_{ij} = G$ for some fixed integer G. Let the column vectors of K represent a primitive set. We will assume for the following three

lemmata that the columns of K have been arranged in lexicographically decreasing order. Let us define $I^*(j)$ as indicating the position of the unique row maximizer for the j^{th} column according to the complete ordering $\underset{i}{>}$. By the assumption that the columns of K form a primitive set, $\{I^*(j)\}$ can be considered a permutation of the integers $(1, 2, \ldots, n)$. We will use the following convention: if $j = 1$, $j - 1 = n$: if $j = n$, $j + 1 = 1$; if $I^*(j) = 1$, $I^*(j) - 1 = n$, and if $I^*(j) = n$, $I^*(j) + 1 = 1$.

As an example of a matrix K whose column vectors form a primitive set, consider the following:

$$\begin{bmatrix} 0 & 0 & -1 & -1 \\ 1 & 1 & 2 & 1 \\ 17 & 16 & 16 & 17 \\ 2 & 3 & 3 & 3 \end{bmatrix}$$

with $G = 20$. The permutation giving the row maximizers for this matrix is $I^*(1) = 1$, $I^*(2) = 4$, $I^*(3) = 2$, and $I^*(4) = 3$. The next four lemmata will prove that this is indeed a primitive set.

Lemma 1: No row of K consists of identical elements.

Proof: Let us assume that all the elements in the i^{th} row are identical and let $I^*(j') = i$ and $I^*(j^*) = i + 1$, i.e., column j' has the row maximizer for row i and column j^* has the row maximizer for row $i + 1$. Now by assumption the $(i + 1)^{th}$ element in column j^* is greater, with respect to $\underset{i+1}{>}$ than the $(i + 1)^{th}$ element in

column j'. Thus since the elements immediately above them in row i are identical, the i^{th} element in column j^* must be greater with respect to $\underset{i}{\geq}$, than the i^{th} element in column j'. But this contradicts the assumption that $I^*(j') = i$, i.e., that column j' had the row maximizer for row i. Therefore no row of K consists of identical elements. Q.E.D.

Lemma 2: In any specific row of K, no two elements can differ by more than one unit.

Proof: Let us look at row i and assume that $k_{ij'} \leq k_{ij^*} - 2$ for some columns j' and j^* where $I^*(j^*) = i$. Now construct a vector $(k_{1j'}, \ldots, k_{i-1,j'} - 1, k_{ij'} + 1, k_{i+1,j'}, \ldots, k_{nj'})'$ in P. This vector will have its m^{th} component smaller (with respect to $\underset{m}{\geq}$) than the greatest (with respect to $\underset{m}{\geq}$) element in row m for all m, which contradicts the assumption that the set K was primitive. To see this, write in juxtaposition the column vectors j', j^*, the new vector, and the vector j where $I^*(j) = m$, starting at row m.

j'	j*	j	New
$k_{mj'}$	k_{mj*}	k_{mj}	$k_{mj'}$
.	.	.	.
.	.	.	.
.	.	.	.
$k_{i-1,j'}$	$k_{i-1,j*}$	$k_{i-1,j}$	$k_{i-1,j'}-1$
$k_{ij'}$	k_{ij*}	k_{ij}	$k_{ij'}+1$
$k_{i+1,j'}$	$k_{i+1,j*}$	$k_{i+1,j}$	$k_{i+1,j'}$
.	.	.	.
.	.	.	.
.	.	.	.
$k_{nj'}$	k_{nj*}	k_{nj}	$k_{nj'}$
$k_{lj'}$	k_{lj*}	k_{lj}	$k_{lj'}$
.	.	.	.
.	.	.	.
.	.	.	.
$k_{m-1,j'}$	$k_{m-1,j*}$	$k_{m-1,j}$	$k_{m-1,j'}$

There are three situations to consider: $m = i-1$, $m = i$ and $m \neq i-1$ or i. In the latter case, where $m \neq i-1$ or i, the first element k_{mj} in column j will, by definition, be greater with respect to $\underset{m}{>}$ than the element $k_{mj'}$ in column j'. Since the constructed vector is identical to the j'^{th} vector down to row $i-1$ and at that point is one unit smaller, then clearly the m^{th} element in the constructed vector is smaller with respect to $\underset{m}{>}$ than the row maximizer k_{mj}. The second case to be considered is where $m = i-1$. In the above juxtaposition of vectors, this situation occurs when row $i-1$ is placed at the top and all the rows above it are displaced to the

bottom. Again $k_{i-1,j}$ is assumed to be the row maximizer, so that $k_{i-1,j}$ is larger with respect to $\overset{>}{i-1}$ than $k_{i-1,j'}$ in the j^{th} vector. But the element $k_{i-1,j'} - 1$ in the $(i-1)^{th}$ position of the constructed vector is strictly smaller numerically than the element $k_{i-1,j'}$ in the j'^{th} vector and is consequently smaller with respect to $\overset{>}{i-1}$ than it. Because the ordering $\overset{>}{i-1}$ is transitive, the element $k_{i-1,j'} + 1$ in the constructed vector is smaller with respect $\overset{>}{i-1}$ than the row maximizer $k_{i-1,j}$. Finally, the last case to be considered is where $m = i$, i.e., where the juxtaposition of vectors begins with the i^{th} row at the top and all the rows above it are displaced to the bottom. In this case column j^* and column j are identical since they both contain the row maximizer for row i. Thus k_{ij^*} equals k_{ij} and since $k_{ij'}$ is numerically smaller than k_{ij} by at least two units, the i^{th} element in the constructed vector is numerically smaller by at least one. Again, because it is numerically smaller $k_{ij'}$ must be smaller with respect to $\overset{>}{i}$. Thus in all cases the i^{th} coordinate in the constructed vector is smaller (with respect to $\overset{>}{i}$) than the row maximizers for all $i = 1, 2, \ldots, n$. This contradicts the assumption that the set K of vectors is a primitive set. Therefore, in any specific row of K, no two elements can differ by more than one unit. Q.E.D.

Lemma 3: All the rows of K are decreasing with respect to $\overset{>}{i}$ in the sense that $k_{ij^*} \overset{>}{i} \cdots \overset{>}{i} k_{in} \overset{>}{i} k_{il} \overset{>}{i} \cdots \overset{>}{i} k_{i,j^*-1}$ for all i, where j^* depends on i.

Proof: The proof will be by induction on i, the row index. First when $i = 1$, row 1 is decreasing with respect to $\geq_i$ by the assumption that the columns of K were arranged in lexicographically decreasing order. Now suppose that the lemma is true for row $m-1$, i.e., row $m-1$ of K is decreasing with respect to the ordering $\geq_{m-1}$. By the assumption that the columns of K form a primitive set, this row must have a row maximizer at some column, say column j. Thus by Lemma 1 and Lemma 2 we have $k_{m-1,j} = k_{m-1,j+1} = \cdots = k_{m-1,j^*-1}$ and $k_{m-1,j^*-1} - 1 = k_{m-1,j^*} = \cdots = k_{m-1,n} = \cdots = k_{m-1,j-1}$ for some j^*. The elements in row m below each set of equal elements in row $m-1$ must be decreasing with respect to $\geq_m$ since row $m-1$ is by assumption decreasing with respect to $\geq_{m-1}$. Thus $k_{mj} \geq_m k_{m,j+1} \geq_m \cdots \geq_m k_{m,j^*-1}$ and $k_{mj^*} \geq_m \cdots \geq_m k_{mn} \geq_m \cdots \geq_m k_{m,j-1}$. Note that we need not consider the situation where $k_{m,j^*-1} \geq_m k_{mj^*}$ for then column j would contain the row maximizer for the m^{th} row. But since by assumption column j already contains the row maximizer for row $m-1$ this contradicts the assumption that the columns of K formed a primitive set. To prove that row m is strictly decreasing with respect to $\geq_m$ we need only show that $k_{m,j-1} \geq_m k_{mj}$. By transitivity of the ordering $\geq_i$, if this is true, column j^* will contain the row maximizer for row m. Assume then that $k_{mj} \geq_m k_{m,j-1}$. Construct a vector in P from column $j-1$ whose i^{th} component is strictly smaller than the maximizer (with respect to $\geq_i$) of the corresponding i^{th} row for all $i = 1, 2, \ldots, n$. Specifically, this vector is $(k_{1,j-1}, \ldots, k_{m-2,j-1} - 1, k_{m-1,j-1} + 1, k_{m,j-1}, \ldots, k_{n,j-1})$. To

see this note that all the components of this new vector except the $(m-1)^{th}$ are smaller with respect to the ordering $\underset{i}{\geq}$ than the components of column $j-1$, which are in turn less with respect to $\underset{i}{\geq}$ than (in one component it is equal to) the corresponding row maximizers. Hence, except possibly for the $(m-1)^{th}$ position, the constructed vector is less with respect to $\underset{i}{\geq}$ than the corresponding row maximizers. Now by Lemma 1 and Lemma 2 $k_{m-1,j-1}$ is numerically one unit less than $k_{m-1,j}$. Thus $k_{m-1,j-1} + 1$ is numerically equal to $k_{m-1,j}$.

Consequently in order to compare these two components with respect to the ordering $\underset{m-1}{>}$ one must compare the components immediately below them in row n. But by assumption $k_{mj} \underset{m}{\geq} k_{m,j-1}$; thus $k_{m-1,j-1} + 1 \underset{m-1}{\leq} k_{m-1,j}$. Now $k_{m-1,j}$ is the row maximizer for row m-1. Thus all the components of the new vector are smaller with respect to $\underset{i}{\geq}$ than the corresponding row maximizers, which contradicts the assumption that the set of column vectors of K were primitive. Thus the assumption that $k_{mj} \underset{m}{\geq} k_{m,j-1}$ is false, which demonstrates that the components of row m are strictly decreasing with respect to $\underset{m}{\geq}$ starting at column j*. Therefore all the rows of K are decreasing with respect to $\underset{i}{\geq}$ in the sense that when $I^*(j^*) = i$, $k_{ij^*} \underset{i}{\geq} \cdots \underset{i}{\geq} k_{in} \underset{i}{\geq} k_{il} \underset{i}{\geq} \cdots \underset{i}{\geq} k_{i,j^*-1}$ for all $i = 1, 2, \ldots, n$. Q.E.D.

The following lemma characterizes primitive sets taken from the set P as defined above.

Lemma 4: Let K be an $n \times n$ matrix and I^* be the permutation on the n first positive integers defined above. Then the columns of K represent the vectors of a primitive set, if and only if, there exists a permutation $I^*(j)$ of the integers $(1, 2, \ldots, n)$ and a rearrangement of the columns of K such that the j^{th} column of K is identical with column $j-1$ except for the entry in row $I^*(j)$, which is one unit larger, and the entry in row $I^*(j)-1$ which is one unit smaller.

Proof: The necessity of the conditions follows immediately from the previous three lemmata. Indeed, by rearranging the columns of K in lexicographically decreasing order, Lemma 3 has shown that the m^{th} row of K is decreasing with respect to $\underset{m}{\geq}$ starting at some column j^* where $I^*(j^*) = m$. In the proof of Lemma 3 the components of row $m-1$ were shown to decrease by one unit at column j^*, i.e., $k_{m-1,j^*-1} - 1 = k_{m-1,j^*}$. Since $I^*(j^*) = m$, the m^{th} component in column j^* must be one unit greater than the m^{th} component in column $j^* - 1$, i.e., $k_{m,j^*-1} + 1 = k_{mj^*}$. but this is precisely the conditions of the lemma.

The sufficiency of the conditions will be demonstrated by induction on m, the size of the matrix K. If K has only one row, the lemma is trivially true. Instead of demonstrating directly that the truth of the lemma for K of size $m-1$ implies its truth for K of size m, we will prove the contrapositive. Specifically, we will assume that the conditions of the lemma hold for a matrix K of size m

which does not represent a primitive set of column vectors; we will then prove that the derived matrix K'' of size $m-1$ which satisfies the conditions of the lemma cannot represent a primitive set of column vectors.

Let K be a matrix of size m which satisfies the conditions of the lemma. Let $k = (k_1, k_2, \ldots, k_m)$ be a vector in P which prevents the column vectors of K from representing a primitive set, i.e., $k_i \underset{i}{\leq} \max_j(k_{ij})$. Since $k \in P$, k must have integral entries such that $k_i \geqslant Q_i$ and $\sum_{i=1}^{m} k_i = G$, the unique column sum of K.

Now k_i must be equal to the row maximum with respect to the ordering $\underset{i}{\geq}$, i.e., $k_i = \max_j(k_{ij})$ for some i. Otherwise, comparing k with an arbitrary column vector of K, say the q^{th}, we have: $k_i \leqslant \max_j(b_{ij}) - 1 \leqslant k_{iq}$. But if $i = I^*(q)$, $\max_j(k_{ij}) - 1 < k_{iq}$ so that $G = \sum_{i=1}^{m} k_i < \sum_{i=1}^{m} k_{iq} = G$: a contradiction.

Suppose then that $k_p = \max_j(k_{pj})$. Construct from the vector k a new vector $k' = (k_1, k_2, \ldots, k_{p-1}, k_p + k_{p+1}, k_{p+2}, \ldots, k_m)$ of one less dimension than k. Derive the $(m-1) \times m$ matrix K' from the matrix K by a similar construction for each column vector of K. Every component of k' is less with respect to the ordering $\underset{i}{\geq}$ than the corresponding components of each column vector in the reduced matrix K' of dimension $(m-1) \times m$. To see this, write in juxtaposition the column vector k' and an arbitrary column vector of K' with the i^{th} row on top, where $I^*(j) = i$.

k'	Column vector j
k_i	k_{ij}
k_{i+1}	$k_{i+1,j}$
.	.
.	.
.	.
k_{p-1}	$k_{p-1,j}$
k_p+k_{p+1}	$k_{pj}+k_{p+1,j}$
k_{p+2}	$k_{p+2,j}$
.	.
.	.
.	.
k_m	k_{mj}
k_1	k_{1j}
.	.
.	.
.	.
k_{i-1}	$k_{i-1,j}$

Since k_i was originally less with respect to the ordering $\geq_i$ than k_{ij}, it is because $k_t < k_{tj}$ for some t where $t = 1, 2, \ldots, m$, i.e., the tie is broken between some corresponding components of k' and the column vector j of K. If the tie were originally broken before or at row $p-1$, it would still be broken as before, since the components before the $(p-1)^{th}$ have not been modified in the construction. A similar reasoning applies if the tie were originally broken at or after row $p+2$. Hence, the only components that could be of concern are the p^{th} and the $(p+1)^{th}$ components and correspondingly

the p^{th} component of the new vectors made up of the sum of the p^{th} and $(p+1)^{th}$ components. But, since $k_p = \max_j(k_{pj})$, $k_p \geq k_{pj}$ so that if the tie had not been broken up the component $(p-1)$, $k_p = k_{pj}$, and the tie cannot be broken at the p^{th} component either. If then the tie were originally broken at the $(p+1)^{th}$ component, it would now be broken by the sum of the p^{th} and the $(p+1)^{th}$ component. Consequently, every component of k' is less with respect to the ordering $\underset{i}{\geq}$ than the corresponding components of each column vector in the reduced matrix K' of dimension $(m-1) \times m$. This matrix will appear as follows:

$$\begin{matrix}
k_{11} & \cdots & k_{1j} & \cdots & k_{1m} \\
\vdots & & \vdots & & \vdots \\
k_{p-1,1} & \cdots & k_{p-1,j} & \cdots & k_{p-1,m} \\
k_{p1}+k_{p+1,1} & \cdots & k_{pj}+k_{p+1,j} & \cdots & k_{pm}+k_{p+1,m} \\
k_{p+2,1} & \cdots & k_{p+2,j} & \cdots & k_{p+2,m} \\
\vdots & & \vdots & & \vdots \\
k_{m1} & \cdots & k_{mj} & \cdots & k_{mn}
\end{matrix}$$

such that every component of the vector k' in P is smaller with respect to the ordering $\underset{i}{\geq}$ than the corresponding row maximizer.

Now consider the column j^* such that $I^*(j^*) = p+1$. Both the $(j^*-1)^{th}$ and the j^{*th} columns of K' are identical. To see this

recall that K satisfies the conditions of the lemma which state that two adjacent columns will differ only in two rows. In this case column j^* and j^*-1 of K will differ only in rows p and $p+1$. Thus k_{p+1,j^*} will be one unit larger than k_{p+1,j^*-1} and k_{pj^*} will be one unit smaller than k_{p,j^*-1}. Hence, the sum of the p^{th} and the $(p+1)^{th}$ components will be identical in column j^*-1 and in column j^*. Consequently the matrix K'' derived from K' by omitting column j^* satisfies the conditions of the lemma; it does not represent a primitive set of column vectors. Since the size of K'' is $m-1$, the induction hypothesis is proved. Therefore, the conditions of the lemma are sufficient to characterize primitive sets given the complete ordering $\gtrsim_i$. Q.E.D.

Lemma 5: Let K represent a primitive set as defined in Lemma 4. Then the unique vector $(k'_{1j}, \ldots, k'_{nj})'$ defined by:

$$k'_{ij} = k_{ij} - 1 \quad \text{for} \quad i = I^*(j) \quad \text{and} \quad i = I^*(j+1) - 1$$

$$k'_{ij} = k_{ij} + 1 \quad \text{for} \quad i = I^*(j) - 1 \quad \text{and} \quad i = I^*(j+1)$$

and

$$k'_{ij} = k_{ij} \quad \text{otherwise;}$$

is the unique replacement for column j. If $I^*(j) = I^*(j+1) - 1$, then $k'_{ij} = k_{ij} - 2$ for this common value of i, and if $I^*(j) - 1 = I^*(j+1)$, $k'_{ij} = k_{ij} + 2$ for the common row. If any $k'_{ij} < Q_i$ then there is no unique replacement from P which will form a primitive set with the remaining vectors after the column vector j is removed.

Proof: All that is needed is to show that the new matrix satisfies the conditions of Lemma 4, since the replacement technique will always generate a unique vector. By Lemma 4 any two adjacent columns of a primitive set will differ in only two rows. Hence if one compares three columns j-1, j and j+1, they will differ by at most four rows, namely I*(j)-1, I*(j), I*(j+1)-1, and I*(j+1). Assuming that these are distinct the submatrix will appear as follows:

$$\begin{array}{c} \\ I^*(j+1)-1 \\ I^*(j+1) \\ I^*(j)-1 \\ I^*(j) \end{array} \begin{array}{c} \begin{array}{ccc} j-1 & j & j+1 \end{array} \\ \begin{bmatrix} a+1 & a+1 & a \\ b & b & b+1 \\ c+1 & c & c \\ d & d+1 & d+1 \end{bmatrix} \end{array}$$

where a, b, c, and d are integers bounded from below by their respective Q_i.

The replacement technique stated in the lemma will modify column j and produce a new submatrix as follows:

$$\begin{array}{c} \begin{array}{ccc} j-1 & j & j+1 \end{array} \\ \begin{bmatrix} a+1 & a & a \\ b & b+1 & b+1 \\ c+1 & c+1 & c \\ d & d & d+1 \end{bmatrix} \end{array}$$

This new matrix satisfies the conditions of Lemma 4 with a new permutation I*' defined by I*'(j) = I*(j+1) and I*'(j+1) = I*(j) and equal to I* otherwise.

When $I^*(j)-1 = I^*(j+1)$ the three columns differ in only three rows. In this case the submatrix becomes:

$$\begin{array}{c} \\ I^*(j+1)-1 \\ I^*(j+1) \\ I^*(j) \end{array} \begin{array}{c} \begin{array}{ccc} j-1 & j & j+1 \end{array} \\ \begin{bmatrix} a+1 & a+1 & a \\ b+1 & b & b+1 \\ c & c+1 & c+1 \end{bmatrix} \end{array}$$

In this case the replacement technique will produce a new submatrix as follows:

$$\begin{array}{c} \begin{array}{ccc} j-1 & j & j+1 \end{array} \\ \begin{bmatrix} a+1 & a & a \\ b+1 & b+2 & b+1 \\ c & c & c+1 \end{bmatrix} \end{array}$$

which also satisfies the conditions of Lemma 4 with the same permutation given above.

The last possibility is when $I^*(j) = I^*(j+1) - 1$. In this case the submatrix becomes:

$$\begin{array}{c} \\ I^*(j)-1 \\ I^*(j) \\ I^*(j+1) \end{array} \begin{array}{c} \begin{array}{ccc} j-1 & j & j+1 \end{array} \\ \begin{bmatrix} a+1 & a & a \\ b & b+1 & b \\ c & c & c+1 \end{bmatrix} \end{array}$$

which the replacement technique changes to:

$$\begin{array}{c} \begin{array}{ccc} j-1 & j & j+1 \end{array} \\ \begin{bmatrix} a+1 & a+1 & a \\ b & b-1 & b \\ c & c+1 & c+1 \end{bmatrix} \end{array}$$

As before, this satisfies the conditions of Lemma 4 with the same permutation, unless $b = Q_{I*(j)}$; then there is no unique replacement from P which will form a primitive set with the remaining vectors after the column vector j is removed. Q.E.D.

The replacement operation defined in Lemma 5 is reversible in the sense that if the vector which was the unique replacement is removed from the resultant primitive set, its unique replacement is precisely the vector which was removed originally.

We noted above that for the applications in this paper the elements of P_k would be taken from the unit hyperplane. The lemmata one through five have characterized the set K and the unique replacement of primitive sets from P for a set P from the G-hyperplane, i.e., the components of the vectors summed to G.

Note that the sign of the integer G has never been specified in any of the preceding lemmata, and that the proofs do not depend on it. This implies that primitive sets can also be defined on hyperplanes whose components sum up to a negative value of G. We wish to show, moreover, that for any positive or negative rational number H, one can define primitive sets from the elements of P_k on the H-hyperplane by an appropriate mapping of primitive sets from the G-hyperplane, where G is any integer positive or negative. In order to prove this, we will first demonstrate an alternative definition of primitive sets taken from the set P defined on the G-hyperplane.

Lemma 6: Let P be the set of all vectors in n-dimensional Euclidean space of the form $(k_1, \ldots, k_n)$ where each k_i is an integer bounded from below by some integer Q_i and such that $\sum_{i=1}^{n} k_i = G$ where G is an integer. A set K of n column vectors from P represents the vectors of a primitive set, if and only if, every column vector of K contains exactly one row minimizer according to the ordering $\geq_i$ and there is no vector belonging to the set P which is greater (with respect to the ordering $\geq_i$) in all components than the row minimizer (with respect to the ordering $\geq_i$) of the set K.

Proof: To prove Lemma 6 we will show that the conditions of the Lemma will provide a characterization of the columns of the matrix K that is equivalent to the characterization of primitive sets obtained in Lemma 4, and that these conditions are sufficient for such a characterization. The proof will proceed in six steps.

For the first three steps we will assume that the columns of K satisfy the conditions of the Lemma and that they have been arranged in lexicographically increasing order. $I(j)$ will be defined to be the row of K which contains the single row minimizer (with respect to the ordering $\geq_i$) for column j.

1. No row of K consists of identical elements. This follows immediately from the proof of Lemma 1 by making the following changes: replace the symbol "I*" with "I", the words "maximizer" with "minimizer" and "greater" with "less."

2. In any specific row of K, no two elements can differ by more than one unit. This follows immediately from the proof of Lemma 2 by making the following changes: replace the symbol "I*" with "I," the words "maximizer" with "minimizer," "greater" with "smaller," "smaller" with "greater," and "greatest" with "smallest"; change the assumption $k_{ij} \leq k_{ij*} - 2$ to $k_{ij} \geq k_{ij*} + 2$; and replace the constructed vector with $(k_{ij'}, \ldots, k_{i-1,j'} + 1, k_{ij'} - 1, k_{i+1,j'}, \ldots, k_{nj'})$.

3. All the rows of K are increasing with respect to $\geq_i$ in the sense that $k_{ij*} \leq_i \cdots \leq_i k_{in} \leq_i k_{il} \leq_i \cdots \leq_i k_{i,j*-1}$ for all i where $j*$ depends on i. This follows immediately from the proof of Lemma 3 by making the following changes: replace the symbol "I*" with "I," the words "decreasing" with "increasing," "maximizer" with "minimizer," "smaller" with "larger," and "less" with "greater"; change the direction of all inequalities; everywhere that a 1 is added to a component, subtract a 1 and everywhere that a 1 is subtracted from a component, add a 1.

4. Let K be an $n \times n$ matrix and let I be the permutation on the n first positive integers defined above. Then the columns of K satisfy the conditions of Lemma 6, if and only if, there exists a permutation $I(j)$ of the integers $(1, 2, \ldots, n)$ and a rearrangement of the columns of K such that the j^{th} column of K is identical with column $j-1$ except for the entry in row $I(j)$, which is one unit smaller, and the entry in row $I(j) - 1$ which is one unit larger.

As was shown in the proof of Lemma 4, the necessity of these conditions follows immediately from the first three parts of this proof. The proof of the sufficiency of these conditions also follows immediately from the sufficiency proof in Lemma 4 by making the following changes: replace the symbol "I*" with "I," the words "maximum" with "minimum," "less" with "greater," "larger" with "smaller," and "maximizer" with "minimizer," change the direction of all inequalities; and change the expression "$\max_k(k_{ij}) - 1$" to read "$1 + \min_j(k_{ij})$," and the expression "$\max_j(k_{ij})$" to read "$\min_j(k_{ij})$."

5. The characterization of every set K of n column vectors from P which satisfies the conditions of Lemma 6 is equivalent to the characterization of primitive sets in Lemma 4. Indeed, let K be a primitive set as in the proof of Lemma 4. Then the columns of K are arranged in lexicographically decreasing order. Moreover, row I*(j) and I*(j) - 1 for any two columns j and j-1 of K appear as follows:

$$\begin{array}{c} \\ I^*(j)-1 \\ I^*(j) \end{array} \begin{array}{c} \begin{array}{cc} j-1 & j \end{array} \\ \begin{bmatrix} a+1 & a \\ b & b+1 \end{bmatrix} \end{array}$$

Now if the columns of K are rearranged in lexicographically increasing order, any such adjacent two columns will simply be interchanged and appear as follows:

$$\begin{array}{c} \\ I(j)-1 \\ I(j) \end{array} \begin{array}{c} \begin{array}{cc} j-1 & j \end{array} \\ \begin{bmatrix} a & a+1 \\ b+1 & b \end{bmatrix} \end{array}$$

where the permutation I^* indexing row maximizers has been exchanged for the permutation I indexing row minimizers. Since any two columns of the set K in Lemma 4 and any two columns of the set K in Lemma 6 differ only in two components, the rearrangement above shows clearly that the set K in Lemma 4 is identical to the set K in Lemma 6 except for the lexicographical ordering in the opposite direction. Consequently, the characterization of any set K of n column vectors from P which satisfies the conditions of Lemma 6 is equivalent to the characterization of primitive sets in Lemma 4.

6. Both characterizations by indexing permutations as described in Lemma 4 and Lemma 6 are necessary and sufficient for the set K to satisfy their corresponding set of conditions respectively. Hence the two sets of conditions are equivalent. Consequently, since the set K in Lemma 4 is a primitive set, the set K in Lemma 6 is also a primitive set. Therefore, a set K of n column vectors from P represents the vectors of a primitive set, if and only if, every column vector of K contains exactly one row minimizer according to the ordering $\geq_i$ and there is no vector belonging to the set P which is greater (according to the ordering $\geq_i$) in all components than the row maximizer (with respect to the ordering $\geq_i$) of the set K. Q.E.D.

It is interesting to note that with primitive sets defined in terms of row minimizers, the replacement operation is completely analogous to that of Lemma 5. Specifically, replace I^* with I and reverse the operation of addition and subtraction in the statement of Lemma 5.

We will now demonstrate that the elements of a set P_k can be taken from any hyperplane which is parallel to the G-hyperplane, where the components of a vector of the hyperplane sum to any rational number H.

Lemma 7:

Let 1. H be a rational number which will identify the hyperplane from which the elements of a set P_k should be taken in the sense that if $p^j \in P_k$, then $\sum_{i=1}^{n} p_i^j = H$.

2. G be an integer whose absolute value will determine the number of elements k of P_k.

3. P be the set of all vectors in n-dimensional Euclidean space of the form $(k_1, \ldots, k_n)$ where each k_i is an integer bounded from below by some integer Q_i and such that $\sum_{i=1}^{n} k_i = G$. We will say that if $p \in P$, p is on the G-hyperplane.

4. $\lambda = H/G$ be a rational number.

Then the mapping of $(k_1, \ldots, k_n)$ into $(\lambda k_1, \ldots, \lambda k_n)$ of the elements of P into the elements of P_k will preserve the primitive sets as characterized by Lemma 4 and will preserve the unique replacement operation on primitive sets as characterized by Lemma 5.

Proof: The mapping of $(k_1, \ldots, k_n)$ into $(\lambda k_1, \ldots, \lambda k_n)$ of an element on one hyperplane to an element in a parallel hyperplane is an isomorphism for any rational number as can easily be verified.

In order to demonstrate that the results of Lemma 4 and Lemma 5 are carried over to elements of P_k taken from the H-hyperplane, all that need be demonstrated is that this mapping preserves primitive sets.

Specifically, let P be a set of vectors as defined in the Lemma and let K be a matrix whose columns are vectors from P and such that they form a primitive set with respect to the ordering $\geq_i$. It must be shown that the columns of the matrix λK form a primitive set from P_k with respect to the same ordering.

Let K be such a matrix whose columns form a primitive set from P with respect to the ordering $\geq_i$. Then, by definition, every column of K contains exactly one row maximizer with respect to the ordering $\geq_i$ and there is no vector from P which is smaller (with respect to the ordering $\geq_i$) than the row maximizer with respect to the ordering $\geq_i$ in all components. Furthermore, by Lemma 6, every column of K contains exactly one row minimizer and there is no vector from P which is greater (with respect to the ordering $\geq_i$) than the row minimizer with respect to the ordering $\geq_i$. Now λK must be a primitive set from P_k, i.e., each column of λK must contain exactly one row maximizer and there must not exist any vector from P_k which is smaller (with respect to the ordering $\geq_i$) than the row maximizer with respect to the ordering $\geq_i$ in all components. In order to see this assume that the column vectors of λK do not form a primitive set, i.e., either there is a column vector from λK which does not have a row maximizer with respect to the ordering $\geq_i$ or else there exists a vector $p^j \in P_k$ such that $p^j_i \leq \max_i \{\lambda k_{ij} \mid j=1, \ldots, n\}$ for all i. First assume that $\lambda > 0$. If there is a column of λK which does not have a row maximizer, then the corresponding column of K will not have a row maximizer, which contradicts the assumption that the

columns of K formed a primitive set. Thus there must exist a $p^j \in P_k$ such that $p_i^j \leq \max_i \left\{ \lambda k_{ij} \mid j=1, \ldots, n \right\}$ for all i. But then $p_i' = \frac{p_i^j}{\lambda} \leq \max_i \left\{ k_{ij} \mid j=1, \ldots, n \right\}$ for all i so that the column vectors of K' do not form a primitive set which again contradicts the assumption that the columns of K did form a primitive set. Now assume that $\lambda < 0$. If there is a column of λK which does not have a row maximizer, then the corresponding column of K will not have a row minimizer which contradicts the assumption that the columns of K formed a primitive set. Again if the column vectors of λK do not form a primitive set it must be that $p_i^j \leq \max_i \left\{ \lambda k_{ij} \mid j=1, \ldots, n \right\}$ for all i. But then $p_i' = \frac{p_i^j}{\lambda} \geq \min_i \left\{ k_{ij} \mid j=1, \ldots, n \right\}$ for all i which again contradicts the assumption that the column vectors of K formed a primitive set. Thus the column vectors of λK must form a primitive set and the mapping $(k_1, \ldots, k_n)$ into $(\lambda k_1, \ldots, \lambda k_n)$ of the elements on the G-hyperplane to elements on the H-hyperplane preserves primitive sets. Clearly, by a similar reasoning, the inverse mapping from P_k to P also preserves primitive sets. Q.E.D.

We have thus shown an alternative formulation of the elements in P_k which are readily generated algebraically. We now have the machinery which is necessary to state and prove the central theorem of the paper.

The Fundamental Theorem:

Let 1. P_k be the set of all vectors in n-dimensional Euclidean space of the form $(\lambda k_1, \ldots, \lambda k_n)$ where each k_i is an integer

bounded from below by some integer Q_i, G is an integer such that $\sum_{i}^{n} k_i = G$, and $\lambda = H/G$ so that if $p^j \in P_k$, $\sum_{i=1}^{n} p_i = H$.

2. B be a matrix whose j^{th} column corresponds to the j^{th} column of P_k. This correspondence will be arbitrary except when an element of P_k contains a λk_i such that $k_i = Q_i$ (a boundary element of the set P_k). In this case the vector in B will be the unit vector with a one at the position of the first such k_i.

3. w be a nonnegative vector in n-dimensional space.

Assume that the set of nonnegative vectors x satisfying the system of equations $Bx=w$ is bounded. Then there exists a primitive set $p^{j1}, p^{j2}, \ldots, p^{jn}$ from P_k as characterized by Lemma 4 and Lemma 7 such that the columns $j_1, j_2, \ldots, j_n$ of B form a feasible basis for the system $Bx=w$.

Proof: Any primitive set as characterized by Lemma 4 will serve as a starting primitive set which has $n - 1$ of its vectors each having a component on a boundary so that these $n - 1$ vectors are associated with $n - 1$ different unit vectors, and the remaining vector, call it p^{k*}, of the primitive set has no component on the boundary. Geometrically such a situation can be seen in Figure 1. Any of the little triangles marked with an S will have their vertices satisfying the above criteria. Consequently finding a starting primitive set from P_k will always be possible and also very easy. Also the columns 1 to n of the matrix B will form a feasible basis for $Bx=w$, with column k not associated with p^{k*}.

There are $n-1$ columns of B both in the feasible basis and uniquely associated with columns of a primitive set from P_k at the beginning of the algorithm and the technique of the algorithm will be to alternate between simplex pivots and the replacement operation on the primitive set described in Lemma 5 so that at any intermediate stage there will remain $n-1$ columns of B both in the basis and associated with the primitive set from P_k. The algorithm will terminate at the time that all the columns of B which are in the basis form a nonnegative solution to $Bx=w$ and are also associated with a primitive set in P_k. This is, of course, the statement of the Fundamental Theorem. So what remains to be shown is that such a position will be reached.

At the outset it should be noted that the assumption that the set of nonnegative vectors x satisfying the system of equation $Bx=w$ is bounded, guarantees that at any stage of the algorithm any vector not in the feasible basis can be brought into the feasible basis by a simplex type pivot. Indeed, the only circumstance in which it is not possible to carry out a simplex pivot is when there is no positive element in the column vector to be introduced into the basis, i.e., no pivot element can be found. Suppose at some stage of the algorithm the j^{th} column of B contained only nonpositive elements. This implies $b_j = \sum_{i=1}^{n} b_{ij}\bar{b}_i$ where $(\bar{b}_i)_{1\leq i\leq n}$ is the current basis and $b_{ij} \leq 0$ for all $i = 1, 2, \ldots, n$. Now $w = \sum_{i=1}^{n} w_i\bar{b}_i$ in terms of that basis so that $w = \sum_{i=1}^{n} w_i\bar{b}_i + x_j(b_j - \sum_{i=1}^{n} b_{ij}\bar{b}_j)$ for arbitrary

$x_j > 0$. Thus $w = \sum_{i=1}^{n} (w_i - x_j b_{ij})\bar{b}_i - x_j b_j$ is a nonnegative solution to $Bx=w$ for any $x_j > 0$ since $(w_i - x_j b_{ij}) > 0$ for $i = 1, 2, \ldots, n$. Since $x_j > 0$ is arbitrary the set of nonnegative solutions x to $Bx=w$ is unbounded. Thus if it is not possible to carry out a simplex pivot, the set of solutions x to $Bx=w$ is unbounded. Consequently, the boundedness of nonnegative solutions x to $Bx=w$ guarantees that a simplex pivot can be carried out at any stage of the algorithm.

At the starting position there is only one possibility for the algorithm, namely to introduce column k^* of B into the basis. There is no replacement in P_k for the vector p^{k^*} since this is the special case referred to in Lemma 5, i.e., when $k'_{ij} < Q_i$. Now when column k^* of B is introduced in the basis a unique vector b^d will drop out of the basis. If $d = k$ the algorithm would terminate since each of the vectors of the primitive set would be associated with columns of B which form a feasible basis for the nonnegative solution to $Bx=w$. If this is not the case the algorithm will drop p^d out of the primitive set and some unique vector p^{d^*} will be introduced into the primitive set. At this stage only p^{d^*} of the primitive set is not associated with a column of B in the feasible basis. The algorithm thus alternates between simplex pivots and the unique replacement operation on primitive sets and always guarantees that it is the unit vector with a one at the k^{th} position which is in the feasible basis but is not associated with a vector in the primitive set, until the algorithm terminates.

At every intermediate stage of the algorithm there are only two possibilities. Either a column of B is introduced replacing a column currently in the basis or else a vector in the primitive set is withdrawn and a unique replacement from P_k is added. The operations are alternated so that if a particular position is reached by a simplex pivot on the B matrix then the next step of the algorithm is a replacement operation on the primitive set. On the other hand, if the particular position is reached by a replacement operation on a primitive set, then the next step of the algorithm is a simplex pivot on the B matrix.

The unique replacement vector generated by the procedure of Lemma 5 will always be from the set P_k itself, that is, we will never reach a position which is the exceptional case in Lemma 5 above, i.e., when $k'_{ij} < Q_i$. Only in the last case presented in the proof of Lemma 5 could a replacement vector be such that it contains a $k'_{ij} < Q_i$. This would occur if $b = Q_{I^*(j)}$. For convenience we will rewrite the matrix for this case when $I^*(j) = I^*(j+1) - 1$ and we are removing column j.

$$\begin{array}{c} \\ I^*(j)-1 \\ I^*(j) \\ I^*(j+1) \end{array} \begin{array}{c} \begin{array}{ccc} j-1 & j & j+1 \end{array} \\ \begin{bmatrix} a+1 & a & a \\ b & b+1 & b \\ c & c & c+1 \end{bmatrix} \end{array}$$

What we wish to show is that this situation with b a boundary element could never occur in the algorithm of the Fundamental Theorem. While this argument will be carried out for primitive sets defined from the

set P on the G-hyperplane, by Lemma 7 this argument applies to primitive sets defined from the set P_k of this corollary. Specifically we will assume that b is a boundary element and that this situation was arrived at in the algorithm of the Fundamental Theorem, and show that this leads to a contradiction.

At the outset we should recall that columns j-1, j and j+1 are identical except in the rows listed and that any column with a component on the boundary will be associated with the unit vector with a one at the position of the first or highest boundary element coordinatewise.

Since b is a boundary element it must be the first boundary element in column j-1, for if it were not, column j-1 would have another boundary element in some row above row I*(j)-1, since (a+1) cannot be a boundary element. Suppose that the highest such boundary element appeared in row p where $p < I^*(j)-1$. Since column j+1 is identical to column j-1 in row p, column j+1 must also have the same first boundary element. Thus both column j+1 and j-1 will be associated with the same unit vector in the basis after column j is replaced, which is impossible in the algorithm of the Fundamental Theorem. Thus b is the first boundary element in column j-1.

Now if a is not a boundary element, we have a similar problem. The b element in column j+1 will be the first boundary element and again column j+1 and j-1 will be associated with the same unit vector. Thus, since by assumption this situation has been arrived at by the algorithm of the Fundamental Theorem, the element a must be a boundary

element. This means that both column j and column j+1 are associated with the same unit vector. However, column j is being replaced so that column j+1 must have just been introduced. Had column j+1 not just been introduced then column j+1 in its present form and column j would have both been associated with the same unit vector and this unit vector would have been in the basis twice, and this is clearly impossible.

Thus we must examine the situation before the previous numbers in column j+1 were modified by the technique of Lemma 5 to produce the current column j+1. In this case we have:

$$
\begin{array}{rc}
 & \begin{array}{cccc} j-1 & j & j+1 & j+2 \end{array} \\
\begin{array}{r} I^{*\prime}(j+1)-1 \\ I^{*\prime}(j+1) \\ I^{*\prime}(j)-1 \\ I^{*\prime}(j+2)-1=I^{*}(j) \\ I^{*\prime}(j+2) \end{array} &
\begin{bmatrix} e+1 & e+1 & e & e \\ d & d & d+1 & d+1 \\ a+1 & a & a & a \\ b & b+1 & b+1 & b \\ c & c & c & c+1 \end{bmatrix}
\end{array}
$$

where we have written $I^{*\prime}(j+1) \neq I^{*\prime}(j) - 1$.

We have used the permutation $I^{*\prime}$ to note that at this position of the algorithm some of the columns (specifically j and j+1) will contain different row maximizers than in the original positions listed. Now $I^{*\prime}(j+1)$ need not be different from $I^{*\prime}(j)-1$ and we will consider this case without rewriting the entire submatrix. If $I^{*\prime}(j+1) = I^{*\prime}(j)-1$ then the row containing the d's and the row containing the a's are identical. Rewriting in terms of the a's the four elements of the row $I^{*\prime}(j+1) = I^{*\prime}(j)-1$ are (a+1), (a),

(a+1), and (a+1). If e is not a boundary element in row I*'(j+1)-1, then the b in column j+2 will be the first boundary element and column j+2 will be associated with the same unit vector as is column j-1. But this is impossible so that in the case where I*'(j+1) = I*'(j)-1 the e in row I*'(j+1)-1 must be a boundary element.

In order to see that the matrix above represents the situation before column j+1 was modified, apply Lemma 5 to column j+1 and obtain the following submatrix:

	j-1	j	j+1	j+2
I*(j+2)-1	e+1	e+1	e+1	e
I*(j+2)	d	d	d	d+1
I*(j)-1	a+1	a	a	a
I*(j+1)-1=I*(j)	b	b+1	b	b
I*(j+1)	c	c	c+1	c+1

which is the original situation with the appropriately added rows and columns. Note that if I*'(j+1) = I*'(j)-1 the row containing the d's could be dropped and the four elements in the row containing the a's would be changed to (a+1), (a), (a), (a+1). Again this reflects the starting situation for this row.

There are three cases to be considered in the situation before column j+1 has been modified. The first situation is when the column j, j+1, and j=2 differ in four rows. This is the case illustrated in the matrix above. The other two situations occur when these three columns differ in only three rows, i.e., when I*'(j+2) = I*'(j+1)-1 or when I*'(j+2)-1 = I*'(j+1). We will treat these last two cases

first. First if $I^*'(j+2) = I^*'(j+1)-1$ the row depicted by e's and the row depicted by c's are identical. If $I^*'(j+1) \neq I^*'(j)-1$ as depicted in the matrix the component a in column $j+2$ and j will be the first boundary element and both columns will be associated with the same unit vector in the feasible basis, which is impossible. Furthermore if $I^*'(j+1) = I^*'(j)-1$ then the b component in column $j+2$ and $j-1$ will be the first boundary element, and again, this is impossible in the algorithm of the Fundamental Theorem. Hence $I^*'(j+2)$ cannot equal $I^*'(j+1)-1$. Also $I^*'(j+2)-1 = I^*'(j+1)$ is not possible since we know that $I^*'(j+2)-1 = I^*'(j)$, which would then imply that $I^*'(j) = I^*'(j+1)$ which is impossible by the definition of I^*' as a permutation indexing the row maximizers. Indeed, no two columns of a primitive set can have the same row maximizer. Thus it must be that the three columns j, $j+1$, and $j+2$ will differ in four rows as depicted in the matrix above.

Given the matrix above depicting the situation before row $j+1$ was modified with the four distinct rows in columns j, $j+1$, and $j+2$, we will show that $I^*'(j+1)-1 < I^*'(j)-1$, i.e., that row $I^*'(j+1)-1$ is above row $I^*'(j)-1$ and that the element e is a boundary element. Note first that d cannot be a boundary element since b is the first boundary element in column $j-1$. Now if $I^*'(j+1) \neq I^*'(j)-1$ as depicted in the matrix and if $I^*'(j+1)-1$ were greater than $I^*'(j)-1$, i.e., if row $I^*'(j+1)-1$ were below row $I^*'(j)-1$, or if the e component in row $I^*(j+1)-1$ were not a boundary element, then the a component in column $j+2$ and j would be the first

boundary element in both columns. By the same argument as given above this is impossible. Also if $I^{*'}(j+1) = I^{*'}(j)-1$ and row $I^{*'}(j+1)-1$ were below row $I^{*'}(j)-1$ and the e component were not a boundary element, then the b component in columns $j+2$ and $j-1$ would be the first boundary elements, which is impossible.

Consequently e must be a boundary element and $I^{*'}(j+1)-1 < I^{*'}(j)-1$, and since all the columns differ only in the elements shown and b is the first boundary element in column $j-1$, the e must be the first boundary element in column $j+1$ and $j+2$, and both of these columns must be associated with the same unit vector. Arguing as before, since column $j+1$ is being removed, it must be that in the previous iteration of the algorithm of the Fundamental Theorem the current column $j+2$ was entered by means of Lemma 5.

Moving back one more step to the situation before column $j+2$ was modified we have:

	$j-1$	j	$j+1$	$j+2$	$j+3$
$I^{*''}(j+2)-1$	g+1	g+1	g+1	g	g
$I^{*''}(j+2)$	f	f	f	f+1	f+1
$I^{*''}(j+1)-1$	e+1	e+1	e	e	e
$I^{*''}(j+1)$	d	d	d+1	d+1	d+1
$I^{*''}(j)-1$	a+1	a	a	a	a
$I^{*''}(j)=I^{*''}(j+3)-1$	b	b+1	b+1	b+1	b
$I^{*''}(j+3)$	c	c	c	c	c+1

where the matrix has been written with $I^{*''}(j+2) \neq I^{*''}(j+1)-1$. Now the argument is exactly analogous to the previous iteration when column $j+1$ was being modified to show that g must be a boundary element in

row $I^{*"}(j+2)-1$, which must be above row $I^{*"}(j+1)-1$. Column $j+2$ and $j+3$ are then both associated with the same unit vector and column $j+3$ must have been introduced in the previous iteration of the algorithm of the Fundamental Theorem.

At some stage of retracing the steps of the algorithm of the Fundamental Theorem, since there are only a finite number of columns in a primitive set, we must conclude that $j+n = j-1$ and that column $j+n$ must have a boundary element in some row above row $I^{*}(j+1)$. But we have shown that if b is a boundary element in row $I^{*}(j+1)$ of column $j+1$, coordinatewise, it must be the first boundary element of column $j-1$. Hence, we have a contradiction and our assumption that b could be an element on the boundary is fallacious.

Under no circumstances then, can the replacement operation in the algorithm of the Fundamental Theorem lead to a new vector outside the boundaries which define the set P_k.

Many of the column vectors of the matrix B can be identical, i.e., the functional mapping from the elements of P_k onto the set of column vectors of B need not be one to one. This will not cause any problem in the algorithm for if a particular column vector is in the basis and the algorithm attempts to introduce the identical vector into the basis, then the column vector which was in the basis originally will be removed. This is easy to see since a vector in the basis at some stage of the algorithm will be represented by a unit vector with a one at some particular row. All identical vectors will also be represented

by the same unit vector so that the algorithm will simply introduce a unit vector into the basis. The pivot row will clearly be the row which contains the one and since the vector is already in the form of a unit vector, the remaining columns of B will not be affected. The pivot row will be divided by one and the nonpivot rows will have a zero element added to each column. Since this simplex pivot operation does not affect the matrix B, it need not be carried out in practice if there is an efficient way of keeping track of which vectors are in the basis. It will suffice to remove from the primitive set the vector which is associated with the identical column of B under consideration. There is no harm in actually performing the simplex pivot step, except for the possibility of introducing unneeded rounding errors into the computations when performing the simplex pivot on the matrix B.

The algorithm cannot cycle in the sense that it cannot return to the same primitive set and consequently to the same n vectors of B, $n-1$ of which are part of a feasible basis. This is because at the start of the algorithm there is only one operation that the algorithm can perform, namely to introduce column k^* of B into the basis with the result that the column vector b^d will be dropped from the basis, i.e., that the column d from the primitive set will not be associated with a vector in the feasible basis. Since this operation produces a unique result and since both the simplex pivot and the replacement operation are reversible, the only way to arrive at the starting position is by means of the situation which occurs when b^d

is not in the basis but b^{k*} is, and b^d is being introduced. But this will clearly not happen at the second step of the algorithm since the next operation will be a replacement operation on P_k, in particular removing p^d from the primitive set. Thus, unless some intermediate stage of the algorithm were repeated, the program could never return to the starting position.

Any intermediate stage of the algorithm is reached either by a replacement operation on primitive sets or a simplex pivot. If it is reached by a simplex pivot then the next step will be a replacement operation on primitive sets and vice versa. Since both of these operations are reversible there are exactly two ways of reaching a particular position in the algorithm, the way the position was reached, and the reverse of the operation leading out of that position. Now if any intermediate stage were to recur, it would imply that there was some third situation which would lead to the intermediate stage. But this is clearly impossible due to the uniqueness of both the simplex pivot and the replacement operation on primitive sets. Consequently, the algorithm cannot cycle.

Furthermore, there are a finite number of primitive sets which can be formed from P_k, a finite set of k vectors, and a finite set of feasible bases to the problem $Bx=w$. Hence, the algorithm must terminate in a finite number of steps. Q.E.D.

In order to guarantee that there is a solution provided by the Fundamental Theorem, the assumption was made that the set of nonnegative

solutions to $Bx=w$ is bounded. No condition has been specified directly on the matrix B which will guarantee that this assumption will be met. For applications of the Fundamental Theorem it is important that the matrix B be appropriately formulated. The following lemma provides the required necessary and sufficient condition on the matrix B.

Lemma 8: The set of nonnegative solutions x to $Bx=w$ is bounded if and only if the column vectors of B are positively linearly independent.

Proof: First if the column vectors of B are positively linearly dependent, then there exists a vector $y \geq 0$ such that $By=0$. But then for any scalar $\lambda > 0$, $B\lambda y=0$. Now if $Bx=w$, it is also true that $Bx + B\lambda y = w$ which says that $(x+\lambda y)$ is a solution for any $\lambda > 0$. Thus since at least one coordinate of y is strictly greater than zero, the set of nonnegative solutions to $Bx=w$ is unbounded.

Conversely, let S be the set of nonnegative solutions to $Bx=w$, let $A(S)$ be the asymptotic cone of S, and assume that S is unbounded. Now S is a closed convex set and, since $x \in S$ is nonnegative, $S \subset \Omega$. Thus $A(S) \subset A(\Omega)$. By the assumption that S is unbounded we have that $A(S) \neq \{0\}$. It can be proved that for any $x \in R^n$, $A(S) = A(S - \{x\})$ and also that if $x \in S$ and $S - \{x\}$ is closed, convex and contains the origin, then, $A(S) = A(S - \{x\}) \subset S - \{x\}$.

Now since $A(S) \neq \{0\}$, there exists $y \in A(S)$, $y \geq 0$. Thus $y \in A(S) \subset (S - \{x\})$ for any $x \in S$. Furthermore, $\lambda y \in A(S) \subset (S - \{x\})$ for all $\lambda > 0$ so that $(x+\lambda y) \in S$ for all $\lambda > 0$. Now fix λ and

choose a $\lambda' > \lambda$. We have that both $B(x+\lambda y) = w$ and that $B(x+\lambda' y) = w$. Thus $B(x+\lambda' y) - B(x+\lambda y) = B(\lambda'-\lambda)y = 0$. Let $z = (\lambda'-\lambda)y$. Now $z \geq 0$ since $(\lambda'-\lambda) > 0$ so that $Bz = 0$ is a nonnegative linear combination of the columns of B which equal zero. Hence the column vectors of B are positively linearly dependent. Q.E.D.

In many applications, the Fundamental Theorem with Degeneracy is readily programmed for digital computers. In particular, if the column vectors of the matrix B associated with the elements of the set P_k can be calculated from the elements of P_k when required, then there is no need to store in the memory of the computer the entire matrix B. By the use of the revised simplex method, only those columns of B associated with the unit vectors need be stored in the computer. When a column of B is to be entered into the basis at some stage of the algorithm, then that vector is transformed into the representation of the current basis by multiplying the vector by the matrix in the columns of original unit vectors. It is this transformed vector which is entered into the basis by means of the conventional linear programming pivot. Correspondingly, the elements of P_k need not all be stored in the computer. In fact all that need be stored at any one time is the current primitive set. The unique replacement vector can be generated algebraically and can be stored in the same position as the vector it replaced.

The set P_k in this algebraic formulation is defined on a specific hyperplane H (in this paper $H = 1$ in all the applications) simply by defining G, the grid size and all the Q_i, the boundaries. In

a particular application one might have some a priori knowledge where on the H-hyperplane the solution will be found. There are two computational techniques which will greatly enhance the efficiency of this algorithm. First, leaving the Q_i at their true values (determined by the particular application of the algorithm), one may start the algorithm at the starting primitive set closest to the assumed answer. In Figure 1 such starting primitive sets are the little triangles marked with an S. This technique has the advantage that it will always find a true solution to the problem in the sense that the algorithm will never terminate on an artificial boundary. The second technique is to pick some computational boundaries Q_i which are greater than or equal to the true boundaries. By choosing larger Q_i's, one essentially chooses a smaller set P_k in the geometric sense and one which has fewer elements (given the same value of G). If the algorithm terminates at a primitive set whose vectors do not contain an element on a computational boundary, then this result will usually be the same as if the true boundaries had been used. This must be qualified since it is conceptually possible for some problem to have more than one solution. By restricting the area of search, one might arrive at a different but true solution. However, if the algorithm terminates at a primitive set whose elements contain any component on a computational boundary, then the solution will not be a true solution. The solution must be recalculated lowering sufficiently the computational boundary on which the previous solution fell so that the final primitive set will not contain any component on a computational boundary.

Application of the Fundamental Theorem

To the General Equilibrium Model

It was shown in Section IV that only relative prices need be investigated for determining the equilibrium price vector. We will restrict our attention to those relative price systems which sum to unity. Consequently we will define P_k for the Fundamental Theorem by setting $H = 1$, G to some appropriate level depending on the degree of precision desired in the result, and $Q_i = 0$ for all i. With $\lambda = 1/G$, the boundary elements of the set P_k are then $\lambda Q_i = 0$. Since λQ_i forms a lower bound for the i^{th} price, the assumption that $Q_i = 0$ is equivalent to the assumption that price i will be nonnegative. The associated matrix B in the Fundamental Theorem will be constructed as follows. As is required in all applications of the Fundamental Theorem the column of B associated with any $p \in P_k$ which has an element λQ_i, in this case zero, will be the unit vector with a one at the position of the first λQ_i. We will interpret the negative of this unit vector to be the disposal activity which disposes of the i^{th} commodity. The column j of B associated with a $p^j \in P_k$ which does not have an element on the boundary Q_i is constructed as follows.

Take $p^j \in P_k$ and compute the profit of each production process at this set of prices. Recall from Section IV that for those production processes which are represented as either a single output Cobb-Douglas or C.E.S. production function this is a two step procedure. First the optimum activity vector must be determined at the price system p^j

then the profits calculated with this optimum activity vector. In the case that the production process is already specified as an activity, the optimum input-output vector is simply this activity vector. Now if the profits associated with the production process which maximizes profits is strictly positive, enter into column j of B the negative of the optimum activity vector. On the other hand, if the maximum profit is negative or zero, calculate the demands for each consumer at the prices p^j, and enter the market demand vector in column j of B, i.e., the sum of the individual demands.

By construction, there is a one-to-one correspondence between the vectors in P_k and the vectors in B. Now, let w be the nonnegative vector of resources initially owned by all the consumers in the economy. Since the B matrix is composed of optimal activity vectors and nonnegative demand vectors, the assumption that the vectors in the production sector are positively linearly independent presented in Section IV guarantees that the set of nonnegative solutions to $Bx=w$ is bounded. Indeed, if the set of nonnegative solutions is unbounded, by Lemma 8, the column vectors of B would be positively linearly dependent, i.e., there would exist a nonnegative, nonzero vector x' such that $Bx' = 0$. Now this solution can be rewritten $Ix^{1'} + B'x^{2'} + B''x^{3'} = 0$ where I is the identity matrix, the negative of the disposal activities, B' is a matrix composed of only the demand vectors in B, and B'' is a matrix composed of only the optimal activity vectors in the production sector. Factoring out I from the left two terms we obtain $I[x^{1'} + B'x^{2'}] + B''x^{3'} = 0$ and let

$z = [x^{1\prime} + B'x^{2\prime}]$ which is clearly a nonnegative vector. Let $A' = [I, B'']$ be the matrix of the negative of the optimal activity vectors in B and let $z' = [z, x^{3\prime}]$ be a nonnegative vector. Then from above $A'z' = 0$. Consequently $-A'z' = 0$ so that the columns of A are positively linearly dependent and production is reversible: a contradiction. Hence, all the conditions of the Fundamental Theorem are verified. Therefore, there exists a primitive set $p^{j1}, p^{j2}, \ldots, p^{jn}$ in P_k such that the columns $j_1, j_2, \ldots, j_n$ form a feasible basis for the system $Bx=w$.

If we let a_{ik} be the i^{th} element in the optimal activity vector of the k^{th} productive process and $D_i(p)$ be the market demand for commodity i at price system p, the basic feasible solution to the system $Bx=w$ is given explicitly by:

$$(1) \qquad -\sum_j a_{ik}x_j + \sum_j D_i(p^j)y_j = w_i \quad \text{for all} \quad i = 1, 2, \ldots, n$$

where the index j is a more convenient labeling of the index j_i, x_j is the positive level at which the k^{th} productive activity having the largest strictly positive profit at prices p^j is to be operated, and y_j is the positive weight given to the demand functions whenever at prices p^j the largest profit derived from the productive activities is either negative or zero.

Since each of the p^j's corresponding to positive a_j's or y_j's belong to a primitive set, the p^j's are relatively close to each other. In fact, the larger the number of price vectors selected from the unit simplex as potential equilibirum prices, i.e., the larger the value of

G, the closer the price vectors in the primitive set. Thus by refining ever more the grid of potential prices on the unit simplex, a common price vector p is obtained in the limit. Note that by construction of the matrices P_k and B if one of the first n columns of B, which corresponds to a disposal activity, is included in the solution, say the i^{th} column, the corresponding price p_i will be zero.

Some average of the price vectors in the primitive set can then serve as a good approximation to the common price vector p. If the demand functions are continuous in a small neighborhood of the primitive set, equation (1) then becomes:

$$(2) \qquad -\sum_j a_{ik}x_j + (\sum_j y_j)D_i(p) = w_i \quad \text{for all} \quad i = 1, 2, \ldots, n$$

and the following relations hold:

(a) $\sum_i p_i a_{ik} \leq 0$ for all k whenever $y_j > 0$ for some j;

(b) $\sum_i p_i a_{ik} \geq 0$ whenever $x_j > 0$ for j corresponding to the k^{th} productive activity, including disposal activities.

It is clear from (2) and (b) above that if the y_j's are all equal to zero $-\sum_j a_{ik}x_j = w_i$ for all $i = 1, 2, \ldots, n$ and $\sum_i p_i a_{ik} \geq 0$ for all k for which $x_j > 0$ so that $0 \leq \sum\sum_{ji} p_i a_{ik} x_j = -\sum_i p_i w_i$. This last relation leads to a contradiction if by assumption $w_i > 0$ for all $i = 1, 2, \ldots, n$. Indeed a set of elements from P_k each with a component on a boundary, i.e., each with a zero component, cannot be primitive. Thus $p_i > 0$ for at least one i so that $\sum_i p_i w_i > 0$: a contradiction. The assumption $w_i > 0$ for all $i = 1, 2, \ldots, n$

can be relaxed to $w_i > 0$ for all $i = 1, 2, \ldots, n$. Indeed, since a primitive set from P_k cannot have all of its elements associated with slack vectors, at least one of the positive x_j's corresponds to a productive activity which is not a disposal activity whenever all the y_j's are equal to zero. But then for this productive activity, say the k^{th} activity, $0 < \sum_i p_i a_{ik}$ by continuity of the supply functions so that, while $\sum_i p_i w_i \geq 0$, $0 < \sum\sum_{ji} p_i a_{ik} x_j = -\sum_i p_i w_i$: a contradiction. Therefore $y_j > 0$ for at least one j. It follows immediately from (a) and (b) above that $\sum_i p_i a_{ik} = 0$ whenever $x_j > 0$.

Multiply each term in (2) by p_i and add over the index i to obtain:

$$(3) \qquad (\sum_j y_j) \sum_i p_i D_i(p) = \sum_i p_i w_i$$

If then Walras' law holds, that is, if the consumers spend all of their income, which can only be derived in this economy by selling their resources on the market, $\sum_i p_i D_i(p) = \sum_i p_i w_i$ so that $(\sum_j y_j) = 1$. Substituting $(\sum_j y_j) = 1$ in (2) it becomes clear that supply equals demand in all the commodity markets.

Thus at some average of the price vectors in the primitive set each consumer maximizes his utility from which process the demand functions are derived, all productive processes which are operated at a positive level yield a maximum profit equal to zero, and all commodity markets are cleared. Therefore general equilibrium is attained and approximate equilibrium prices are computed.

In any actual computation of equilibrium prices the algorithm terminates with a primitive set of prices. An average of these prices is needed in order to calculate the equilibrium supply and demand. For an averaging of these prices Scarf has chosen a technique of using linear programming to find that average price which minimizes the maximum deviation between demand and supply. Specifically he formulates the linear program:

Maximize $$x_1 + x_2 + \cdots + x_n$$

Subject to:

$$\begin{aligned}
d_{11}x_1 + \cdots + d_{1n}x_n - a_{11}x_{n+1} - \cdots - a_{1m}x_{n+m} &\leq w_1 \\
d_{21}x_1 + \cdots + d_{2n}x_n - a_{21}x_{n+1} - \cdots - a_{2m}x_{n+m} &\leq w_2 \\
&\vdots \\
d_{n1}x_1 + \cdots + d_{nn}x_n - a_{n1}x_{n+1} - \cdots - a_{nm}x_{n+m} &\leq w_n \\
p_{11}x_1 + \cdots + p_{1n}x_n &\leq 0 \\
p_{21}x_1 + \cdots + p_{2n}x_n &\leq 0 \\
&\vdots \\
p_{m1}x_1 + \cdots + p_{mn}x_n &\leq 0
\end{aligned}$$

where $d_{ik} = D_{ik} - \left(\sum_{i=1}^{n} D_{ij} \;/\; (n+1) \right)$, where D_{ij} is the demand for commodity i at the price system p^j on the final primitive set and where $p_{ik} = P_{ik} - \left(\sum_{i=1}^{n} P_{ik} \;/\; (n+1) \right)$ where P_{ij} is the profit of activity i at the same price system p^j.

This linear program produces a weight to be applied to each price in the final simplex. Let $V = \sum_{i=1}^{n} x_i$ be the value of the linear program. Then $\frac{x_i}{V/(n+1)} - 1$ is the weight to apply to the i^{th} price. The sum of these weights is clearly unity.

An alternative averaging scheme has been programmed which is computationally faster and appears to give better results when computing equilibrium prices in the allocation model. This method finds the barycenter of the points in the final primitive set of prices, i.e., it calculates the unweighted average of the price vectors.

Application of the Fundamental Theorem To the Allocation Model

Section IV has described how to each allocation of money resources imposed by the government there corresponds a set of equilibrium prices in the regional economy. The problem for the government is to decide which allocation of the money resources will result in a set of equilibrium prices in the regional economy which are as close as possible to the set of relative prices in the larger economy for those commodities common to both economies.

The solution to this problem is a straight forward application of the Fundamental Theorem. Recall that the Fundamental Theorem considers two matrices, P_k and B, in the n-dimensional Euclidean space such that the j^{th} column of B is associated with the j^{th} column of P_k, and a nonnegative vector w such that the set of nonnegative vectors x satisfying $Bx=w$ is bounded. Whenever these conditions

are satisfied, there exists a primitive set $p^{j_1}, \ldots, p^{j_n}$ in P_k such that the columns $j_1, \ldots, j_n$ of B form a feasible basis for $Bx=w$. Recall also that the concept of a primitive set in P_k can be interpreted intuitively as a set of vectors relatively close to each other.

In the present application of the Fundamental Theorem, the matrix P_k will consist of a bounded subset of column vectors on the unit hyperplane derived from a set P from the G-hyperplane, where G is a positive integer, by normalizing the vectors of P to sum to unity. These vectors will represent allocations of the money resources to each commodity common to both economies, i.e., the components of each vector in P_k define the proportion of the money resources to be spent on each commodity common to both economies. The matrix B will consist of column vectors representing price systems and obtained in the following way. Each allocation on a boundary of P_k will be arbitrarily associated with the unit vector with the one in the coordinate which has the first boundary element. Each non-boundary allocation in P_k will be associated with the prices of the corresponding commodities normalized to sum up to unity and computed from the general equilibrium model of the regional economy with the initial vector of physical resources derived from this particular allocation. The vector w will consist of the set of prices in the larger economy corresponding to the commodities common to both economies and normalized to sum up to unity.

Since the column vectors in the matrix B represent either unit vectors or equilibrium price systems, their components are all nonnegative and consequently the columns of B are positively linearly independent. Hence, the set of vectors x such that $Bx=w$ is bounded. All the conditions of the Fundamental Theorem are satisfied. Therefore, there exists a primitive set of allocations in P_k such that a nonnegative linear combination of the equilibrium prices associated with each allocation in the primitive set is equal to the normalized prices in the larger economy, i.e., $\sum_{j=1}^{n} x_j b^j(a^j) = w$ where $b^j(a^j)$ is the price system in B corresponding to the allocation a^j in P_k, and $x_j \geq 0$ for all j.

This linear combination guarantees that the vector of prices in the larger economy will always be in the convex hull of the price systems associated with the allocations in the final primitive set. Indeed, the nonnegative weights x_j must sum up to unity. To see this, recall that each $b^j(a^j) \in B$ and w are normalized to sum up to unity, i.e., $\sum_{i=1}^{n} b_i^j(a^j) = 1$ and $\sum_{i=1}^{n} w_i = 1$. Now $\sum_{j=1}^{n} x_j b^j(a^j) = w$ means that:

$$\sum_{j=1}^{n} x_j b_i^j(a^j) = w_i \quad \text{for all } i = 1, 2, \ldots, n.$$

Thus:

$$\sum_{i=1}^{n} \left(\sum_{j=1}^{n} x_j b_i^j(a^j) \right) = \sum_{j=1}^{n} \left(\sum_{i=1}^{n} b_i^j(a^j) \right) x_j$$

$$= \sum_{j=1}^{n} x_j = \sum_{i=1}^{n} w_i = 1$$

and $\sum_{j=1}^{n} x_j = 1$ as asserted above.

Now, increase the grid size indefinitely to obtain a finer partition of the allocation hypersurface. To each grid size G, there corresponds a primitive set of allocations $\left\{a^{1G}, a^{2G}, \ldots, a^{nG}\right\}$ which converges to a unique allocation a. To each grid size G, there also corresponds a nonnegative linear combination of the price systems $\left\{b^{1G}(a^{1G}), b^{2G}(a^{2G}), \ldots, b^{nG}(a^{nG})\right\}$ associated with the allocations in the primitive set, i.e., $\sum_{j=1}^{n} x_j^G b^{jG}(a^{jG}) = w$. Clearly, the sequence $\left(\sum_{j=1}^{n} x_j^G b^{jG}(a^{jG})\right)_{G \in I}$, where I is the index set of nonnegative integers, converges. Consequently, if for every j the mapping b^j is continuous:

$$\lim_{G\to\infty} \sum_{j=1}^{n} x_j^G b^{jG}(a^{jG}) = \sum_{j=1}^{n} x_j b^j(a) = b(a) = w$$

Indeed, $b^j(a) = b(a)$ for all j and $\sum_{j=1}^{n} x_j = 1$. Therefore, whenever the mapping b^j is continuous for every j, the prices in the regional economy are matched exactly with the prices in the larger economy.

By construction, however, the mapping b is not continuous for any allocation a^j on a boundary of the allocation hypersurface P_k. Indeed, each allocation on a boundary of P_k is arbitrarily associated with the unit vector with the one in the coordinate which has the first boundary element. Thus, prices in the regional economy which correspond to allocations on the boundary of the allocation hypersurface P_k will not be matched exactly with prices in the larger economy.

There is no guarantee that the mappings b^j corresponding to allocations a^j interior to the allocation hypersurface P_k will be

continuous on P_k. The pricing mechanism may indeed by very sensitive to the initial vector of physical resources. However, while the conditions which would guarantee the continuity of b^j are not known, computational experience seems to indicate that the mappings b^j are in fact continuous for allocations interior to the allocation hypersurface.

Assuming again the continuity of the mappings b^j for interior allocations, the prices which correspond to nonboundary allocations will be matched exactly in the sense that they will be proportional to the corresponding prices in the larger economy and hence in the same proportion to each other. To see this let K be the set of indices such that if $j \in K$, a^{jG} will contain a boundary element where a^{jG} is an element in the final simplex of allocations for a grid size G for all $G > N$ where N is fixed and arbitrarily large. Let M be the set of indices indicating the first boundary element for each such a^{jG} where $j \in K$. Then:

$$\lim_{G\to\infty} \sum_j x_j^G b^{jG}(a^{jG}) = \sum_j \lim_{G\to\infty} x_j^G b^{jG}(a^{jG})$$

$$= \sum_{j\in K} \lim_{G\to\infty} x_j^G b^{jG}(a^{jG}) + \sum_{j\notin K} \lim_{G\to\infty} x_j^G b^{jG}(a^{jG})$$

$$= \sum_{j\in K} x_j I^m + \sum_{j\notin K} x_j b^j(a)$$

$$= \sum_{j\in K} x_j I^m + b(a)\left(\sum_{j\notin K} x_j\right) = w$$

since $b^j(a) = b(a)$ for all $j\notin K$ and $m \in M$. Now for $i \notin M$,

$x_j I^i = 0$ so that $b_i(a)\left(\sum_{j \notin K} x_j\right) = w_i$. Thus since M contains the indicies of those prices associated with boundary elements, the prices $b_i(a)$ not associated with boundary allocations are proportional to the prices in the larger economy w_i where $\sum_{j \notin K} x_j$ is the factor of proportionality.

It was indicated above that even in this case the equilibrium allocation will be a Pareto optimal allocation of the money resources. No entrepreneur can take advantage of the price differential for those goods whose prices are associated with boundary allocations. By the assumption that the absorptive capacity has been met, the entrepreneur cannot export more of that commodity to the larger economy. Restricting the exporting of such a commodity will make the regional economy poorer and hence, at least one consumer in the regional economy will be worse off. Thus there will be no way to trade with the commodities associated with boundary allocations. It was shown above that all the remaining commodities have the same relative prices as prevail in the larger economy. Hence, there can be no arbitrage available for the entrepreneur in these commodities. Consequently, even in the case where prices are not exactly matched between the regional economy and the larger economy, the allocation of money resources will be Pareto optimal.

Figure 2 illustrates graphically a hypothetical solution in which at least one price in the regional economy will not be matched exactly with the corresponding price in the larger economy. The allocation hypersurface P_k is the unit simplex of allocations where $Q_i = 0$

for all $i = 1, 2, 3$. The non-boundary allocations map into a subset B of the unit price simplex. The vector w, representing the relative prices in the larger economy, does not belong to the set B. Consider the primitive set determined by the vectors a^1, a^2, and a^3, where a^1 is a boundary element in P_k. The boundary allocation a^1 maps into the unit vector p^1 with the one in the second coordinate on the unit price simplex, i.e., the vertex labeled 2. The algorithm guarantees that for each grid size the relative price vector w in the larger economy will always be in the convex hull of the prices p^1, p^2, and p^3 corresponding to the allocations a^1, a^2, and a^3 in the final primitive set. If the set of prices B on the unit simplex, in which all non-boundary allocations map, contains the relative price vector w, every price in the regional economy will be matched exactly to the corresponding price in the larger economy, i.e., the limiting allocation a will map on the price vector w.

As was already discussed in the third part of Section IV, the situation in which it is impossible to match exactly all the prices in both economies arises for either of two reasons. Either the numerical value of Q_i is too high for one or more i, i.e., there is not sufficient absorptive capacity in the larger economy; or else some productive activity involving all commodities in both economies is profitable at the prices prevailing in the larger economy. Theoretically, the first problem can always be remedied by appropriately increasing the absorptive capacity of the larger economy. It is only the latter situation that can never be remedied since there is no bound

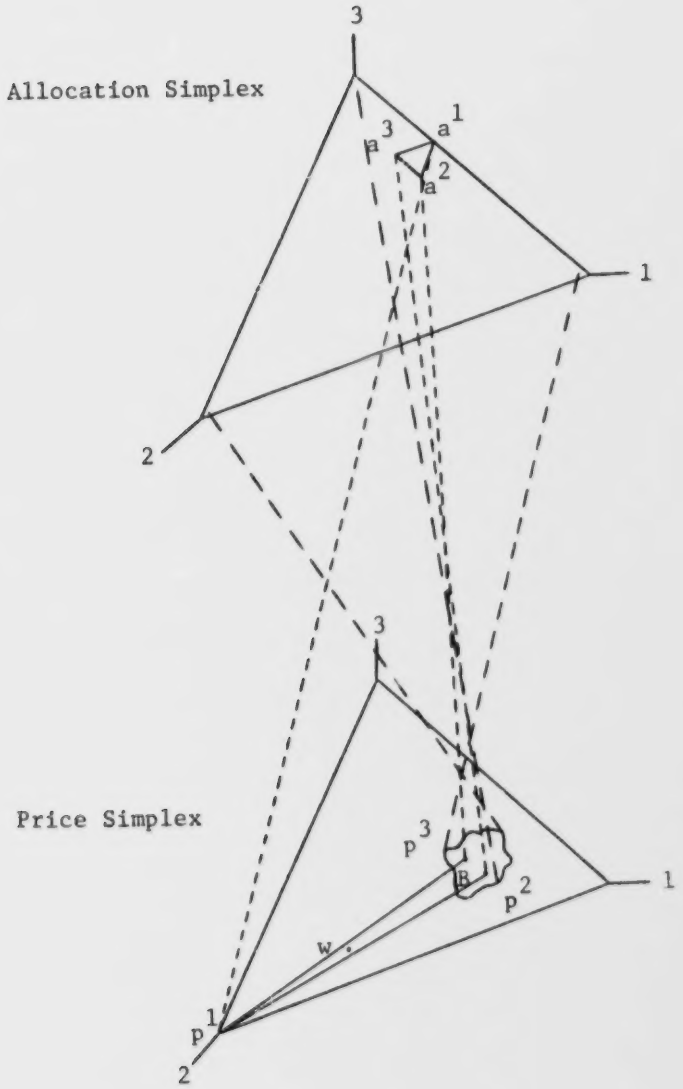

Fig. 2.--A geometric view of the mapping from a set P_k of allocations to the unit simplex of prices relative to some hypothetical allocation model.

to the productive capacity of the regional economy. In practice, however, even the first situation can remain whenever the absorptive capacity of the larger economy is rigidly determined.

In general, it would be computationally prohibitive to define the allocation hypersurface P_k by its "true" boundaries specifying the "true" absorptive capacity of the larger economy. The set P_k would then be very large. For computational convenience, "computational" boundaries are then chosen, each no smaller numerically than the "true" Q_i, and a solution is found. So long as the final simplex of allocations does not contain a vector on a "computational" boundary, the result will be the same as with the lower "true" boundary. Otherwise, the "computational" boundary should be lowered. The procedure consists of choosing the "computational" Q_i's such that the resulting set of allocations P_k will be as small as possible and still map into a subset of the price simplex which contains the vector w of relative prices in the larger economy. This can always be done unless some productive activity involving all commodities in both economies is profitable at the prices in the larger economy. A useful technique is to choose a crude grid size G and solve the model with a moderate size P_k such as setting $Q_i = 0$ for all i. In general, the final primitive set of allocations will have a number of allocations with components on the boundary Q_i. One can then decrease the numerical value of Q_i and compute the model over and over again, increasing the grid size whenever the final primitive set of allocations contains no boundary element. Such a procedure can often increase appreciably the computational efficiency of the allocation model.

An Alternative Method of Solution Of the Allocation Model

The allocation model is a two-stage application of an algorithm which approximates fixed points of a continuous mapping [10]. Its interaction mechanism approximates fixed points of fixed points, i.e., fixed points of the composition of two mappings: a mapping of the money resources into a vector of initial physical resources and a mapping of the initial physical resources into a vector of equilibrium prices. From a computational point of view, the allocation model would be greatly simplified if the composite mapping could be identified.

While this composite mapping has not yet been discovered, it will be shown that a direct application of the general equilibrium algorithm to the regional economy specified in a very particular way provides results identical to those obtained with the allocation model.

In his dissertation, Terje Hansen describes an economy with a foreign trade sector [4]. Specifically, he introduces a new commodity which he calls foreign exchange and two new sets of activities. The first set consists of the import activities, each of which converts some amount of the foreign exchange commodity and of some other commodities into a unit amount of the commodity to be imported. The other set consists of the export activities, each of which produces some amount of foreign exchange by using up one unit of the commodity to be exported and some amount of some other commodities. The general equilibrium algorithm then computes the relative price of foreign exchange as

well as the optimum levels of operation of these import and export activities. If there is no final demand for foreign exchange, all the foreign exchange gained from exports is spent on imports. Since foreign exchnage is used only in trade, the optimum levels of operation of the trade activities determine the optimum allocation of foreign exchange between the commodities imported and exported.

The foreign exchange commodity in the Hansen trade model is exactly analogous to the money resource in the allocation model. The quantity of foreign exchange given up in importing, or gained in exporting, one unit of any traded commodity is precisely the price of the commodity in the world market in terms of foreign exchange. Similarly, the quantity of the money resource used in buying, or gained in selling, one unit of the commodity in the larger economy is precisely the price of the commodity in the larger economy in terms of the money resource. In essence then the trade activities define the prices prevailing in the larger economy and their optimum levels of operation determine the optimum allocation of the money resource to the commodities involved in trade between the regional economy and the larger economy.

The objective of the allocation model is to match as closely as possible the relative prices of the commodities in the regional economy with the corresponding relative prices in the larger economy. A foreign trade sector added on to the general equilibrium model of the regional economy would accomplish an exact matching of relative prices in both economies provided each trade activity involves no other commodity but the traded commodity itself and the money resource. To see

this, consider the following "frictionless" trade activities. Let M_i be the price of commodity i in the larger economy, that is, the amount of the money resource involved in the trade of one unit of commodity i. Then, the frictionless import activity for commodity i is represented by the vector $(0, \ldots, 1, \ldots, -M_i)$ with the 1 in the i^{th} component and its frictionless export activity is represented by the vector $(0, \ldots, -1, \ldots, M_i)$ with the -1 in the i^{th} component. Suppose, for example, the general equilibrium solution says to import commodity r and to export commodity t at the relative prices p_r and p_t, respectively, with p_m the equilibrium price of money. These two trade activities must then just break even, i.e., they must exhibit zero profits. Hence,

$$p_r - p_m M_r = 0 \quad \text{and}$$

$$-p_t + p_m M_t = 0$$

so that $\frac{P_r}{P_t} = \frac{M_r}{M_t}$. Thus the relative prices in both economies are exactly matched.

The problem with this approach is that the algorithm which approximates fixed points does not apply to the regional economy with frictionless trade. Indeed, the set of solutions to the corresponding system Bx=w is then unbounded. This follows immediately from the fact that the frictionless import and export activities for any given commodity i are positively linearly dependent, that is, there exist positive scalars x and y such that $x\cdot(0, \ldots, 1, \ldots, -M_i) + y\cdot(0, \ldots, -1, \ldots, M_i) = (0, \ldots, 0, \ldots, 0)$; indeed, take $x = y$

for any $x > 0$. Since the frictionless import and export activities of a given commodity are positively linearly dependent, they cannot be together in the same basic feasible solution. However, since they are positively linearly dependent, the frictionless import and export activities of a given commodity can be in a non basic feasible solution. Thus, in a regional economy with frictionless trade, the set of solutions to the corresponding system $Bx=w$ is unbounded and the approximation algorithm fails to apply. The algorithm cannot decide whether to import or to export the given commodity.

In order to find the general equilibrium solution of a regional economy with trade whenever bounded solutions do exist, either frictions must be built in the trade activities or some a priori knowledge must be employed to exclude from the specification of the model either the import or the export activity for each commodity which can be traded in the larger economy.

If frictions are built in the trade activities, the relative prices of commodities in the regional economy cannot match the relative prices of the corresponding commodities in the larger economy, except by accident. The discrepancies reflect the "external" costs of trade. Each import and export activity with friction has a negative number $-a_k$ in the k^{th} component representing the amount of commodity k needed to import or to export one unit of the traded commodity. Suppose, for example, the general equilibrium solution says tc import commodity r and to export commodity t at the relative prices p_r and p_t, respectively,

with p_m the equilibrium price of money and p_k the equilibrium price of commodity k needed either to import commodity r or to export commodity t or both. Then,

$$p_r - \sum_k p_k a_k - p_m M_r = 0 \quad \text{and}$$

$$-p_t - \sum_j p_j a_j + p_m M_t = 0$$

so that $\dfrac{p_r - \sum_k p_k a_k}{p_t + \sum_j p_j a_j} = \dfrac{M_r}{M_t}$. Clearly, $\dfrac{p_r}{p_t} = \dfrac{M_r}{M_t}$ if and only if $\sum_k p_k a_k = 0$ and $\sum_j p_j a_j = 0$, that is, the relative prices in the regional economy exactly match the relative prices in the larger economy if and only if the "external" costs of trade are zero in each trade activity; otherwise, the discrepancy reflects the "external" costs of trade.

If a priori information is given concerning which frictionless trade activities optimize the allocation of money resources in the regional economy, the other trade activities could be excluded from the specification of the regional economy with trade. The relative prices in each economy would then be exactly matched. This a priori information can be obtained by trial and error computations with the trade model, excluding at each try the trade vector corresponding to the one which is included in the solution at positive level. This procedure can be very tedious since there are 2^n different combinations to try, where n is the number of commodities which can be traded. The same information can be obtained directly by applying the allocation model.

The trade model with frictionless trade activities is very important from a computational point of view. It is faster to compute than the allocation model and yields a closer approximation for similar grid sizes. A suggestion then is to compute the allocation model with sufficient, even though crude, grid sizes to determine which commodities are to be imported and which commodities are to be exported. Once this information is known, apply the trade model to obtain relatively quickly a more refined estimate of the optimum allocation of the money resources. Whenever it is possible to specify the regional economy with trade activities all having enough friction so that the algorithm can effectively distinguish between import and export activities, the trade model is far superior to the allocation model.

SECTION VI

ACKNOWLEDGMENTS

Rev. J. Eugène Poirier, S.J. and Dr. John M. Harrington, Jr. were Principal and Co-Principal Investigators of this project and co-authors of this report.

Dr. Herbert E. Scarf kindly gave permission to use his computer program for the computation of equilibrium prices.

The support of the project by the Water Quality Office, Environmental Protection Agency and the help provided by Mr. Mark A. Pisano, the Grant Project Officer, who provided a great deal of insight and optimism in the seminal discussions from which this research grew, is acknowledged with sincere thanks.

Preceding page blank

SECTION VII

REFERENCES

1. Debreu, Gerard. Theory of Value. New York: John Wiley & Sons, Inc., 1959.

2. Foley, Duncan K. "Lindahl's Solution and the Core of an Economy with Public Goods." Econometrica, XXXVIII (January, 1970), 66-72.

3. Foley, Duncan K. "Resource Allocation and the Public Sector." Yale Economic Essays, VI (Spring, 1967), 45-98.

4. Hansen, Terje. "On the Approximation of a Competitive Equilibrium." PhD dissertation, Yale University, 1968.

5. Hansen, Terje and Scarf, Herbert. "On the Applications of a Recent Combinatorial Algorithm." Cowles Foundation Discussion Paper No. 272 (April, 1969).

6. Harrington, John M., Jr., "General Equilibrium and Regional Allocation of Money Resources: A Methodology for Implementing General Equilibrium Analysis." PhD dissertation, Georgetown University, 1971.

7. Kuhn, Harold W. "Simplicial Approximation of Fixed Points." Proceedings of the National Academy of Sciences, LXI (1968).

8. Scarf, Herbert. "An Example of an Algorithm for Calculating General Equilibrium Prices." The American Economic Review, LIX (September, 1969), 669-677.

9. Scarf, Herbert. "On the Computation of Equilibrium Prices." Ten Economic Studies in the Tradition of Irving Fisher. New York: John Wiley & Sons, Inc., 1967.

10. Scarf, Herbert. "The Approximation of Fixed Points of a Continuous Mapping." SIAM Journal of Applied Mathematics, XIV (September, 1967), 1328-1343.

11. Scarf, Herbert. "The Core of an N Person Game." Econometrica, XXXV (January, 1967), 50-69.

Preceding page blank

SECTION VIII

APPENDICES

Page No.

A. Mathematical Derivation of Demand Functions 119

Derivation of Demand Function from Generalized C.E.S. Utility Function 119

Derivation of Demand Function from Generalized Cobb-Douglas Utility Function 122

Derivation of Input Demands for Unit Output from a Generalized C.E.S. Production Function 124

Derivation of the Input Demands for Unit Output from a Generalized Cobb-Douglas Production Function 130

B. The Computer Program 135

Technical Notes on the Computer Program 135

A User's Guide to the Computer Program 136

A Listing of the Computer Program 147

Preceding page blank

APPENDIX A

MATHEMATICAL DERIVATION OF DEMAND FUNCTIONS

Derivation of Demand Function from Generalized C.E.S. Utility Function

Given a linear homogeneous utility function for a consumer of the form

$$U(x) = \left(\sum_k (b_k)^{1-a} (x_k)^a \right)^{1/a} .$$

The consumer has initial endowment of commodity k, w_k which at a price system p, forms the "income" $\sum_k p_k w_k$. In order to maximize utility subject to the budget constraint

$$\sum_k p_k w_k = \sum_k p_k x_k .$$

Let

$$L(x) = \left(\sum_k (b_k)^{1-a} (x_k)^a \right)^{1/a} + \lambda \left[\sum_k p_k w_k - \sum_k p_k x_k \right]$$

and take all the first partial derivatives

$$\frac{\partial L(x)}{\partial x_j} = \frac{1}{a} \left(\sum_k (b_k)^{1-a} (x_k)^a \right)^{(1/a)-1} a(b_j)^{1-a} (x_j)^{a-1} - \lambda p_j$$

and

$$\frac{\partial L(x)}{\partial \lambda} = \sum_k p_k w_k - \sum_k p_k x_k .$$

Setting the first partial derivatives equal to zero and simplifying where possible

Preceding page blank

(1) $$\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a} (b_j)^{1-a}(x_j)^{a-1} = \lambda p_j$$

and

(2) $$\sum_k p_k w_k = \sum_k p_k x_k \quad .$$

Dividing both sides of (1) by $(x_j)^{a-1}$ we obtain

$$\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a} (b_j)^{1-a} = \lambda p_j (x_j)^{1-a} \quad .$$

Now dividing both sides by $(p_j)^a$

$$\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a} (b_j)^{1-a}\, p_j^{-a} = \lambda (p_j)^{1-a}(x_j)^{1-a}$$

$$= \lambda (p_j x_j)^{1-a} \quad .$$

Now taking the $1-a$ root of entire equation

$$\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{1/a} (b_j)(p_j)^{-a/(1-a)} = \lambda^{1/(1-a)} p_j x_j \quad .$$

Now summing over all the j's

$$\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{1/a} \sum_k (b_k)(p_k)^{-a/(1-a)} = \lambda^{1/(1-a)} \sum_k p_k x_k$$

$$= \lambda^{1/(1-a)} \sum_k p_k w_k$$

by (2).

Now take both sides to the $1-a$ power

$$\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a}\left(\sum_k b_k p_k^{-a/(1-a)}\right)^{1-a} = \lambda\left(\sum_k p_k w_k\right)^{1-a} .$$

Thus

$$\lambda = \left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a}\left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{1-a}\left(\sum_k p_k w_k\right)^{a-1} . \tag{3}$$

Now substituting (3) into (1) we obtain

$$\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a}(b_j)^{1-a}(x_j)^{a-1}$$

$$= \left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a}\left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{1-a}\left(\sum_k p_k w_k\right)^{a-1} p_j .$$

Dividing both sides by $\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a}$ we obtain

$$(b_j)^{1-a}(x_j)^{a-1} = \left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{1-a}\left(\sum_k p_k w_k\right)^{a-1} p_j .$$

Dividing both sides by $(b_j)^{1-a}$ we obtain

$$(x_j)^{a-1} = \left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{1-a}\left(\sum_k p_k w_k\right)^{a-1} p_j (b_j)^{a-1} .$$

Taking the $a-1$ root

$$x_j = \left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{-1}\left(\sum_k p_k w_k\right)(p_j)^{1/(a-1)}(b_j)$$

$$= \frac{b_j \sum_k p_k w_k}{(p_j)^{1/(1-a)}\sum_k b_k (p_k)^{-a/(1-a)}} .$$

Now let $e = \frac{1}{1-a}$, then

$$x_j = \frac{b_j \sum_k p_k w_k}{(p_j)^e \sum_k b_k (p_k)^{1-e}} .$$

Derivation of Demand Function from Generalized Cobb-Douglas Utility Function

Given a linear homogeneous utility function for a consumer of the form

$$U(x) = A \prod_k (x_k)^{b_k} \quad \text{with} \quad \sum_k b_k = 1$$

The consumer has an initial endowment of commodity k, w_k which at a price system p, forms the "income" $\sum_k p_k w_k$.

In order to maximize utility subject to the budget constraint

$$\sum_k p_k w_k = \sum_k p_k x_k$$

let

$$L(x) = A \prod_k (x_k)^{b_k} + \lambda \left[\sum_k p_k w_k - \sum_k p_k x_k \right]$$

and take all the first partial derivatives

$$\frac{\partial L(x)}{\partial x_j} = A b_j (x_j)^{b_j - 1} \prod_{k \neq j} (x_k)^{b_k} - \lambda p_j$$

and

$$\frac{\partial L(x)}{\partial \lambda} = \sum_k p_k w_k - \sum_k p_k x_k .$$

Setting all partial derivatives equal to zero we obtain

$$(1) \qquad Ab_j (x_j)^{b_j - 1} \prod_{k \neq j} (x_k)^{b_k} = \lambda p_j$$

and

$$(2) \qquad \sum_k p_k w_k = \sum_k p_k x_k \quad .$$

Multiplying both sides of (1) by x_j we obtain

$$(3) \qquad Ab_j \prod_k (x_k)^{b_k} = \lambda p_j x_j \quad .$$

Adding all these equations over j gives

$$A \prod_k (x_k)^{b_k} \sum_j b_j = \lambda \sum_j p_j x_j \quad .$$

Now from (2) and the linear homogeneity property $\sum_j b_j = 1$ we obtain

$$A \prod_k (x_k)^{b_k} = \lambda \sum_k p_k w_k \quad ,$$

which on solving for λ gives

$$(4) \qquad \lambda = \frac{A \prod_k (x_k)^{b_k}}{\sum_k p_k w_k} \quad .$$

Now substituting (4) into (3) gives

$$Ab_j \prod_k (x_k)^{b_k} = \frac{A \prod_k (x_k)^{b_k}}{\sum_k p_k w_k} p_j x_j \quad ,$$

which on simplifying and solving for x_j gives

$$x_j = \frac{b_j \sum_k p_k w_k}{p_j} \quad .$$

Derivation of the Input Demands for Unit Output with a Generalized C.E.S. Production Function

Given a generalized linear homogeneous C.E.S. production function of the form

$$P(x) = \left(\sum_k (b_k)^{1-a} (x_k)^a \right)^{1/a} \quad .$$

There are two methods of determining the input demands for unit output. The first is to minimize the costs $C = \sum_k p_k x_k$ subject to being on the unit isoquant, i.e., $P(x) = 1$. The second method is to assume some cost constraint C; maximize output subject to this cost constraint; determine the level of output so obtained; and then to adjust the input demands to produce unit output. The second method is only possible since the function is linear homogeneous.

Method 1

Let

$$L(x) = \sum_k p_k x_k + \lambda \left[1 - \left(\sum_k (b_k)^{1-a} (x_k)^a \right)^{1/a} \right] \quad ,$$

$$\frac{\partial L(x)}{\partial x_j} = p_j - \lambda \frac{1}{a} \left(\sum_k (b_k)^{1-a} (x_k)^a \right)^{(1/a)-1} (a)(b_j)^{1-a} (x_j)^{a-1} \quad ,$$

and

$$\frac{\partial L(x)}{\partial \lambda} = 1 - \left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{1/a} \quad .$$

Setting all partial derivatives equal to zero and simplifying we obtain

(1) $$p_j = \lambda\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a} b_j{}^{(1-a)}(x_j)^{a-1}$$

and

(2) $$\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{1/a} = 1 \quad .$$

Dividing (1) by $(x_j)^{a-1}$ we obtain

$$(x_j)^{1-a}\, p_j = \lambda\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a} (b_j)^{1-a} \quad .$$

Multiplying both sides by $(p_j)^{-a}$

$$(x_j)^{1-a}(p_j)^{1-a} = \lambda\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a} (b_j)^{1-a}(p_j)^{-a} \quad .$$

Then

$$(x_j p_j)^{1-a} = \lambda\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a} (b_j)^{1-a}(p_j)^{-a} \quad .$$

Taking the 1-a root

$$x_j p_j = \lambda^{1/(1-a)}\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{1/a} b_j (p_j)^{-a/(1-a)} \quad .$$

Summing over all equations

$$\sum_k x_k p_k = \lambda^{1/(1-a)}\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{1/a} \sum_k b_k (p_k)^{-a/(1-a)} \quad .$$

Taking the 1-a power

$$\left(\sum_k x_k p_k\right)^{1-a} = \lambda\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a}\left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{1-a} .$$

Solving for λ

$$(3) \qquad \lambda = \left(\sum_k x_k p_k\right)^{1-a}\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{-(1-a)/a}\left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{a-1} .$$

Substituting (3) into (1)

$$p_j = \left(\sum_k x_k p_k\right)^{1-a}\left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{-(1-a)/a}\left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{a-1}$$

$$\times \left(\sum_k (b_k)^{1-a}(x_k)^a\right)^{(1-a)/a} (b_j)^{1-a}(x_j)^{a-1} ,$$

$$(x_j)^{a-1} = p_j\left(\sum_k x_k p_k\right)^{a-1}\left(\sum_k b_k (p_k)^{-a/(1-a)}\right)^{1-a}(b_j)^{a-1} ,$$

and

$$(4) \qquad x_j = \frac{b_j \sum_k p_k x_k}{(p_j)^{1/(1-a)} \sum_k b_k (p_k)^{-a/(1-a)}} .$$

Now substituting this result into (2)

$$\left\{\sum_j (b_j)^{1-a}\left[\frac{b_j \sum_k p_k x_k}{(p_j)^{1/(1-a)} \sum_k b_k (p_k)^{-a/(1-a)}}\right]^a\right\}^{1/a} = 1 .$$

Now $\sum_k p_k x_k$ and $\sum_k b_k (p_k)^{-a/(1-a)}$ do not depend on j and can thus be brought outside of $\sum_j$. We obtain

$$\frac{\sum_k p_k x_k}{\sum_k b_k (p_k)^{-a/(1-a)}} \left\{ \sum_j (b_j)^{1-a} \left[\frac{b_j}{(b_j)^{1/(1-a)}} \right]^a \right\}^{1/a} = 1 \quad .$$

Thus

$$\sum_k p_k x_k = \frac{\sum_k b_k (p_k)^{-a/(1-a)}}{\left\{ \sum_j (b_j)^{1-a} \left[\frac{b_j}{(p_j)^{1/(1-a)}} \right]^a \right\}^{1/a}} = \frac{\sum_k b_k (p_k)^{-a/(1-a)}}{\left(\sum_k (b_k)^{1-a} (b_k)^a (p_k)^{a/(a-1)} \right)^{1/a}}$$

$$= \frac{\sum_k b_k (p_k)^{a/(a-1)}}{\left(\sum_k b_k (p_k)^{a/(a-1)} \right)^{1/a}} = \left(\sum_k b_k (p_k)^{a/(a-1)} \right)^{1-(1/a)} \quad .$$

Substituting this into (4) we obtain

$$x_j = \frac{b_j \left(\sum_k b_k (p_k)^{a/(a-1)} \right)^{1-(1/a)}}{(p_j)^{1/(1-a)} \sum_k b_k (p_k)^{-a/(1-a)}}$$

$$= \frac{b_j}{(p_j)^{1/(1-a)} \left(\sum_k b_k (p_k)^{a/(a-1)} \right)^{1/a}}$$

$$= \frac{b_j}{(p_j)^e \left(\sum_k b_k (p_k)^{1-e} \right)^{1/a}}$$

where $e = \frac{1}{1-a}$.

Method 2

Since the production function is identical to the generalized C.E.S. Utility Function, we can replace the budget constraint $\sum_k w_k p_k$ with the assumed cost constraint C so that we obtain immediately from the results above that production is maximized when

$$x_j = \frac{b_j C}{(p_j)^{1/(1-a)} \sum_k b_k (p_k)^{-a/(1-a)}} .$$

Now if this is substituted into the production function, the output will be

$$P(x) = \left\{ \sum_j (b_j)^{1-a} \left[\frac{b_j C}{(p_j)^{1/(1-a)} \sum_k b_k (p_k)^{-a/(1-a)}} \right]^a \right\}^{1/a} .$$

Since C and $\sum_k b_k (p_k)^{-a/(1-a)}$ do not depend on j they may be brought outside the $\sum_j$ sign:

$$P(x) = \left\{ \left[\frac{C}{\sum_k b_k (p_k)^{-a/(1-a)}} \right]^a \sum_j (b_j)^{1-a} \left[\frac{b_j}{(p_j)^{1/1-a}} \right]^a \right\}^{1/a}$$

$$= \frac{C}{\sum_k b_k (p_k)^{-a/(1-a)}} \left\{ \sum_j (b_j)^{1-a} \left[\frac{b_j}{(p_j)^{1/(1-a)}} \right]^a \right\}^{1/a} .$$

Thus if each x_j is divided by this output, the resulting output will be unity since the production function is linear homogeneous. Thus for unit output

$$x_j' = \frac{x_j}{P(x)} = \frac{\dfrac{b_j C}{(p_j)^{1/(1-a)} \sum_k b_k (p_k)^{-a/(1-a)}}}{\dfrac{C}{\sum_k b_k (p_k)^{-a/(1-a)}} \left\{ \sum_k (b_k)^{1-a} \left[\dfrac{b_k}{(p_k)^{1/(1-a)}} \right]^a \right\}^{1/a}}$$

$$= \frac{b_j}{(p_j)^{1/(1-a)} \left\{ \sum_k (b_k)^{1-a} \left[\dfrac{b_k}{(p_k)^{1/(1-a)}} \right]^a \right\}^{1/a}}$$

$$= \frac{b_j}{(p_j)^{1/(1-a)} \left\{ \sum_k (b_k)^{1-a} (b_k)^a (p_k)^{-a/(1-a)} \right\}^{1/a}}$$

$$= \frac{b_j}{(p_j)^{1/(1-a)} \left[\sum_k b_k (p_k)^{-a/(1-a)} \right]^{1/a}}$$

$$= \frac{b_j}{\left[(p_j)^{1/(1-a)} \sum_k b_k (p_k)^{a/(a-1)} \right]^{1/a}}$$

$$= \frac{b_j}{\left((p_j)^e \sum_k b_k (p_k)^{1-e} \right)^{1/a}} \quad \text{where} \quad e = \frac{1}{1-a} ,$$

which is the identical result as obtained by the first method.

Derivation of the Input Demands for Unit Output for a Generalized Cobb-Douglas Production Function

Given a generalized linear homogeneous Cobb-Douglas production function

$$P(x) = A \prod_k (x_k)^{b_k} \quad \text{where} \quad \sum_k b_k = 1 \quad .$$

There are two methods of determining the input demands for unit output. The first is to minimize the costs $C = \sum_k p_k x_k$ subject to being on the unit isoquant, i.e., $P(x) = 1$. The second method is to assume some arbitrary cost constraint C; maximize output subject to this cost constraint; determine the level of output so obtained; and then to adjust the input demands to produce unit output. The second method is only possible since the function is linear homogeneous.

Method 1

Let

$$L(x) = \sum_k p_k x_k + \lambda \left[1 - A \prod_k (x_k)^{b_k} \right] \quad ,$$

$$\frac{\partial L(x)}{\partial x_j} = p_j - \lambda A b_j (x_j)^{b_j - 1} \prod_{k \neq j} (x_k)^{b_k} \quad ,$$

and

$$\frac{\partial L(x)}{\partial \lambda} = 1 - A \prod_k (x_k)^{b_k} \quad .$$

Setting partial derivatives equal to zero gives

$$(1) \qquad p_j = \lambda A b_j (x_j)^{b_j - 1} \prod_{k \neq j} (x_k)^{b_k}$$

and

$$A \prod_k (x_k)^{b_k} = 1 \quad . \tag{2}$$

Multiplying both sides of (1) by x_j gives

$$p_j x_j = \lambda A b_j \prod_k (x_k)^{b_k} \quad ,$$

which by (2) reduces to

$$p_j x_j = \lambda b_j$$

or

$$x_j = \frac{\lambda b_j}{p_j} \tag{3}$$

$$(x_j)^{b_j} = \lambda^{b_j} \left(\frac{b_j}{p_j} \right)^{b_j} \quad .$$

Multiplying equations together over j

$$\prod_j (x_j)^{b_j} = \prod_j \lambda^{b_j} \prod_j \left(\frac{b_j}{p_j} \right)^{b_j} \quad .$$

Multiplying both sides by A and noting that $\prod_j \lambda^{b_j} = \lambda^{\sum_j b_j}$

$$A \prod_j (x_j)^{b_j} = A \lambda^{\sum_j b_j} \prod_j \left(\frac{b_j}{p_j} \right)^{b_j} \quad ,$$

which by (2) and $\sum_j b_j = 1$ gives

$$1 = A \lambda \prod_j \left(\frac{b_j}{p_j} \right)^{b_j}$$

and

$$\lambda = \frac{1}{A \prod_j \left(\frac{b_j}{p_j}\right)^{b_j}} = \frac{1}{A \frac{\prod_j (b_j)^{b_j}}{\prod_j (p_j)^{b_j}}}$$

$$= \frac{\prod_j (p_j)^{b_j}}{A \prod_j (b_j)^{b_j}} \quad ;$$

and replacing j with k since it is an arbitrary index

(4) $$\lambda = \frac{\prod_k (p_k)^{b_k}}{A \prod_k (b_k)^{b_k}} \quad .$$

Now placing (4) into (3) gives

$$x_j = \frac{\left(\prod_k (p_k)^{b_k}\right) b_j}{A \left(\prod_k (b_k)^{b_k}\right) p_j} = \frac{b_j \left(\prod_k (p_k)^{b_k}\right)}{p_j A \left(\prod_k (b_k)^{b_k}\right)}$$

$$= \frac{b_j \left(\prod_{k \neq j} (p_k)^{b_k}\right)}{(p_j)^{1-b_j} A \left(\prod_k (b_k)^{b_k}\right)} \quad .$$

Method 2

Since the production function is identical to the generalized Cobb-Douglas utility function, we can replace the budget constraint $\sum_k w_k p_k$ with the assumed cost constraint C so that we obtain immediately from the results above that production is maximized when

$$x_j = \frac{b_j C}{p_j} \quad .$$

Now if this is substituted into the production function the output will be

$$P(x) = A \prod_k \left(\frac{C b_k}{p_k} \right)^{b_k} \quad .$$

Now C does not depend on k and can be brought outside the $\prod_k$

$$P(x) = A C^{\sum_k b_k} \prod_k \left(\frac{b_k}{p_k} \right)^{b_k} \quad ,$$

which since $\sum_k b_k = 1$ give

$$P(x) = AC \prod_k \left(\frac{b_k}{p_k} \right)^{b_k} \quad .$$

Thus if each x_j is divided by this output, the resulting output will be unity since the production function is linear homogeneous.

$$x_j' = \frac{x_j}{P(x)} = \frac{\dfrac{Cb_j}{p_j}}{AC \prod_k \left(\dfrac{b_k}{p_k}\right)^{b_k}} = \frac{Cb_j}{p_j AC \prod_k \left(\dfrac{b_k}{p_k}\right)^{b_k}}$$

$$= \frac{b_j \prod_k (p_k)^{b_k}}{Ap_j \prod_k (b_k)^{b_k}}$$

$$= \frac{b_j \prod_{k \neq j} (p_k)^{b_k}}{(p_j)^{1-b_j} A \prod_k (b_k)^{b_k}} \quad ,$$

which is the identical result as obtained by the first method.

APPENDIX B

THE COMPUTER PROGRAM

Technical Notes on the Computer Program

The computer program is written in the FORTRAN IV language with the technique of dynamically dimensioned arrays. The required memory size will depend on the size of the array X in the main routine (the value of NSPACE). If this size is increased then the program can handle larger problems and will consequently use more computer memory. For example, with NSPACE = 10,000, the program uses about 100k bytes on an IBM 360 Model 40.

The program contains the Fundamental Theorem programmed twice, once for the general equilibrium solution and once for the allocation solution. The subroutines ALLOC and PIVOTA perform the linear programming and the replacement operation on primitive sets respectively for the allocation solution while the subroutines SCARF and PIVOT do the same operations for the general equilibrium solution. The subroutines TERM, DEMAND, and SUPPLY apply only to the general equilibrium solution and perform the final linear programming averaging step, deriving demand functions and deriving supply functions, respectively.

In theory the matrix B will contain k columns, one for each vector of P_k. However, the use of the revised simplex method allows one to store in the computer only n + 1 vectors of B, the n columns associated with the slack vectors and the column associated with w, the right hand side of Bx=w. By multiplying the actual vector of B

to be entered into the basis by the matrix of the columns associated with the slack vectors, one obtains that column of B in the current basis. The pivot row can then be determined directly in the usual fashion using the column associated with w which is already in the current basis. In the computer program the $n + 1$ columns of B are stored as BINV and PBINV for the general equilibrium and the allocation solution respectively.

The index associated with each vector in the current primitive set from P_k is the row number which the associated column of the matrix B pivoted on when being entered into the feasible basis for the nonnegative solution of $Bx=w$. The program keeps track of this index in the vector NAME and LNAME for the general equilibrium and allocation solutions respectively. When each column is associated with a column of B in a basis, then the column associated with w contains the nonnegative weights x which provide the nonnegative solution to $Bx=w$.

A User's Guide to the Computer Program

The computer program is written so that it will either compute the general equilibrium prices for an economy or else compute the optimal allocation of a money resource. All aspects of the model as discussed in this paper have been programmed. All the information relating to the economy must be supplied on a set of cards. For reference purposes, the following is an outline of the cards needed to operate this program, in the order they must be supplied to the program. An asterisk next

to an item means that the card(s) must be supplied only for an allocation run.

A. Header card

B. Commodity Labels

C. Production Labels

D. Consumer Labels

E. Description of the Production Sector

F. Physical Resources

G. Description of the Demand Side of the Economy

H. Boundaries for the General Equilibrium Solution (Optional)

*I. Commodity Selection for the Allocation Solution (Optional)

*J. Boundaries for the Allocation Solution

*K. Absolute Prices in the Larger Economy

*L. Money Resources

After one set of data is read and the model processed, the program will attempt to read another set of data, beginning with the Header Card. If there is none there, the program will terminate.

A. Header Card

The header card is always the first card read by the program in any general equilibrium run or any allocation run. All the numbers punched on this card must be punched without decimal points and moved as far to the right in the field as possible. For example if a model contained 14 commodities, the number 14 must be punched in columns 1 to 5 as far to the right as possible, i.e., in columns 4 and 5. The columns marked

with an asterisk apply to the allocation model only and must be left blank for a general equilibrium run.

The following restriction forms the limit on the size of the problem which can be handled by this program. As in the program let N be the number of commodities, NACT be the number of production processes, including disposal activities, NPLAY be the number of consumers, and LN be the number of commodities involved in the allocation run. Let K1 = (NACT + 1)(2NACT + 1) and let K2 = (2N)(N + 1). If column 45 has a one punched then let K1 = 0. Let K = max(K1, K2). Let NSPACE be the size of the array X which contains all the other arrays. For the program as listed in this Appendix, NSPACE = 10,000. For a general equilibrium run, the size of the numbers punched on the header card is bounded by the following algebraic formula:

$$(11 + 8N + 2NPLAY)N + 6NPLAY + 6NACT + K + 18 \leq NSPACE .$$

For an allocation run the size of the numbers punched on the header card is bounded by the following algebraic formula:

$$(11 + 8N + 2NPLAY)N + 6NPLAY + 6NACT + K + (4 + 2LN)LN + 29 \leq NSPACE .$$

The following is a description of each field on the header card:

Col 1-5 Number of Commodities in economy for general equilibrium run or in regional economy for an allocation run.

Col 6-10 Number of Production Processes including disposal activites.

Col 11-15 Number of Consumers in the economy.

Col 16-20 Grid Size for the general equilibrium model. The program assumes a value of 100 if this field is left blank.

Col 21-25 Starting corner for the general equilibrium model. For a general equilibrium run if this field is left blank the program assumes a value of 1. For an allocation run, if this field is left blank the program will start at that corner closest to the equilibrium price at the previous iteration.

Col 26-30 Maximum number of iterations permitted in the general equilibrium solution. If this field is left blank the program assumes a value of 10,000.

Col 31-35 General equilibrium boundary option. If this field is left blank the program will assume $Q_i = 0$ for all i. If any number is punched in this field the program will read in the Q_i. See H below.

Col 36-40 Tax option. Punch any number in this field if the consumers are to be taxed at a fixed rate for each consumer. The tax rate will appear on a resource card as if it were an additional commodity. The government will always be the last consumer.

Col 41-45 Averaging option for final primitive set of prices. Leave field blank if Scarf's linear programming technique should be used. Punch a 1 (one) if the barycenter average is desired.

*Col 46-50 Number of commodities in the allocation solution. If this field is left blank, the program will assume that this number is the same as the number of commodities in the regional economy as specified in Col 1-5. This number may, however, be specified and less than the number punched in Col 1-5, i.e., some commodities can be localized to the regional economy only. In this case, either the commodities in the allocation solution must be the first commodities in the regional economy and then column 66-70 of the Header card may be left blank or else a number must be punched in Col 66-70 of the Header card and then a card must be supplied indicating which commodities are to be included in the allocation solution. See I below.

*Col 51-55 Grid size for allocation solution. This field must be filled in for any allocation run.

*Col 56-60 Starting corner for allocation solution. If this field is left blank the program will assume a value of 1.

*Col 61-65 Maximum number of iterations permitted in the allocation solution. If this field is left blank the program assumes a value of 10,000.

*Col 66-70 Commodity Selection option for Allocation Model. Punch any number in this field if a card should be read in specifying which commodities should be included in allocation model (see I below). This field may be left blank and no card supplied to the program if the commodities in the allocation model are the first commodities of the regional economy.

*Col 71-72 Print Option. Punching a number from 1 to 10 will control the information printed. If this field is left blank only the final general equilibrium solution will be printed. Punch a 1 for the entire printout of the general equilibrium solution at each iteration. The higher the number the less that will be printed.

B. through D. Label Cards

Each commodity, production process, and consumer must be given an eight character descriptive title (or the appropriate number of blank cards supplied). Each type of label must begin on a new card. For example, the name of the first commodity will appear in Col 1-8 of the first commodity label card, the name of the first productive process will appear in Col 1-8 of the first production process label

card and so on. As many cards of each type as are needed must be supplied. For example, if there were fifteen consumers, one would need two consumer label cards. The first card would contain the labels for the first ten consumers. Card two would contain the labels for the additional five consumers, i.e., only Col 1-40 of card two would be filled in.

The label cards must be inserted in the deck in the order specified immediately following the Header Card.

E. Description of the Production Sector

Each non-slack activity must have one or more cards, the number depending on the number of commodities in the general equilibrium solution, specifying the parameters of the production function. Col 1 of the first card for each production process will have a single digit specifying the type of specification of the production process. The following codes have been programmed.

0 or blank	Activity Analysis
1	Cobb-Douglas Production Function
2	C.E.S. Production Function

Except for this number, all other numbers punched on these cards must have a decimal point punched. Each field on these cards is ten columns wide with the specification digit assumed to take up the entire first ten columns of the first card. Col 71-80 of each card is reserved for identifying the card and any alphabetic or numeric data may be punched. Thus the first card will appear as follows:

Col 1 Specification Digit

Col 2-10 Blank

Col 11-20 First parameter

Col 21-30 Second parameter

.
.
.

Col 61-70 Sixth parameter

Col 71-80 Comment

while the second and subsequent card will appaar:

Col 1-10 Parameter

Col 11-20 Parameter

.
.
.

Col 61-70 Parameter

Col 71-80 Comment

It is suggested that the parameter information be punched left adjusted (with a decimal point punched) i.e., starting in Col 11, 21, 31, etc.

Activity Analysis

The parameters in an activity analysis formulation of production will be specified with the following convention: inputs will be specified as negative numbers and outputs will be specified as positive numbers.

Cobb-Douglas and C.E.S. Formulations

The demand parameters for both the Cobb-Douglas and C.E.S. single output production functions will be specified as positive numbers. In order to identify which variable is the output, -1 will be punched in the parameter field for that commodity. It is possible to represent the production of a public good as formulated in this paper by specifying -1 for all the commodities which will be the public goods. In the derived unit output activity vector the unit output number 1 will be repeated for each such commodity.

The scale factor A for the Cobb-Douglas specification and the constant elasticity of substitution will be punched as a parameter in the position for commodity n + 1. For example, if an economy had four commodities then the elasticity of substitution number would be punched in Col 51-60.

F. Physical Resources

One or more cards (depending on the number of commodities in the general equilibrium solution) must be supplied for each consumer in the model specifying the quantity of each commodity which is available to that consumer at the beginning of the general equilibrium solution. Each commodity is assigned a field of ten columns with the last ten columns in each card reserved for identifying information. For example, the quantity of commodity one is punched in columns 1-10 of the first card while the quantity of commodity nine is punched in columns 11-20 of the second card. Again, all numbers must be punched with a decimal point.

G. Description of the Demand Sector

Each consumer must have one or more cards, depending on the number of commodities in the general equilibrium solution, specifying the parameters of the demand function. Except that all the demand parameters must be nonnegative, these cards are punched with the same format as the production sector. The same three types of functions are available with the same codes, namely demand functions derived from utility functions of the activity type, Cobb-Douglas type, and C.E.S. type.

H. Boundaries (the Q_i) for the General Equilibrium Solution

This card (or cards depending on the number of commodities in the general equilibrium solution) must be included only if some number is punched in Col 31-35 on the Header Card. This card specifies the computational boundaries which define the set P_k from which the price systems are taken. If this card is not supplied, the program will automatically set each $Q_i = 0$. The boundary numbers are punched without decimal points and moved as far to the right in each field as possible. Each field is five columns wide so that, for example, the boundary number for commodity one is punched in Col 1-5, for commodity two in Col 6-10 and so on for as many cards as are necessary.

I. Commodity Selection for the Allocation Solution

This card (or cards depending on the number of commodities in the allocation solution) must be included only if some number is punched in Col 66-70 on the Header Card. Punch on this card the sequence

number of each commodity to be included in the allocation model. There must be exactly the same number of numbers punched as is specified in Col 46-50 on the Header Card. These numbers must be punched in ascending order. The format for this card is exactly the same as in H above.

J. Boundaries (the Q_i) for the Allocation Solution

This card (or cards depending on the number of commodities in the allocation solution) must be included with every allocation run. Punch on this card the computational boundaries which define the set P_k from which the allocations are taken. This card is punched with the same format as in H above.

K. Absolute Prices in the Larger Economy

This card (or cards depending on the number of commodities in the allocation solution) must be included with every allocation run. Punch on this card the absolute prices of the commodities in the larger economy which are included in the allocation run. This card should be punched with the same format as applies to punching physical resources in F above.

L. Money Resources

This card (or cards depending on the number of consumers in the economy) must be included with every allocation run. Punch on this card the quantity of money resources which are available to each consumer. This card should be punched with the same format as applies to punching physical resources in F above.

A Listing of the Computer Program

```
C --- PROGRAM TO ALLOCATE MONEY RESOURCES AND/OR COMPUTE EQUILIBRIUM      ALOC0001
C --- PRICES                                                              ALOC0002
C ---                                                                     ALOC0003
C --- THIS PROGRAM FORMS PART OF THE DISSERTATION OF JOHN M.              ALOC0004
C --- HARRINGTON, JR. WHICH WAS SUBMITTED TO THE GRADUATE SCHOOL OF       ALOC0005
C --- GEORGETOWN UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS    ALOC0006
C --- FOR A PH.D. IN ECONOMICS.                                           ALOC0007
C --- THE GENERAL EQUILIBRIUM PORTION OF THE PROGRAM WITH ACTIVITY TYPE   ALOC0008
C --- PRODUCTION AND C.E.S. TYPE DEMAND FUNCTIONS WAS OBTAINED FROM       ALOC0009
C --- DR. HERBERT A SCARF , COWLES FOUNDATION, YALE UNIVERSITY ON         ALOC0010
C --- JUNE 3, 1969                                                        ALOC0011
C ---                                                                     ALOC0012
C --- ALL OTHER ASPECTS OF THE PROGRAM WERE PROGRAMMED AT GEORGETOWN      ALOC0013
C --- UNIVERSITY BY JOHN M. HARRINGTON, JR.                               ALOC0014
C ---                                                                     ALOC0015
C --- THE DEVELOPMENT OF THIS PROGRAM WAS PAID FOR IN PART BY A GRANT     ALOC0016
C --- FROM THE FEDERAL WATER POLLUTION CONTROL AGENCY OF THE DEPARTMENT   ALOC0017
C --- OF THE INTERIOR.                                                    ALOC0018
C ---                                                                     ALOC0019
      DIMENSION X(10000)                                                  ALOC0020
      COMMON/SPACE/ X                                                     ALOC0021
      COMMON K5,K6,M,NN,MAXI,LM,LNN,LMAXI,IBO,ISEL,KPS,ITO               ALOC0022
     *,INDEX,JOUT,IDXS,IDXS1,IDXS2,NG,LIMDEX,LJOUT,IAVG                  ALOC0023
      COMMON/DIMEN/N,NACT,NPLAY,LN,NP,LNP,NACTNN,NTP,NACTP                ALOC0024
      NSPACE=10000                                                        ALOC0025
      K5=5                                                                ALOC0026
      K6=6                                                                ALOC0027
    1 READ(K5,10,END=1000) N,NACT,NPLAY,M,NN,MAXI,IBO,ITO,IAVG,LN,LM,     ALOC0028
     * LNN,LMAXI,ISEL,KPS                                                 ALOC0029
   10 FORMAT(14I5,5I2)                                                    ALOC0030
      IF(LN.EQ.0) LN=N                                                    ALOC0031
      NP=N+1                                                              ALOC0032
      LNP=LN+1                                                            ALOC0033
      NACTNN=NACT-N                                                       ALOC0034
      NTP=2*NACT+1                                                        ALOC0035
      NACTP=NACT+1                                                        ALOC0036
      I1=1                                                                ALOC0037
C --- W                                                                   ALOC0038
      I2=I1+N                                                             ALOC0039
      I2=(I2/2)*2+1                                                       ALOC0040
C --- D                                                                   ALOC0041
      I3=I2+N*NACT                                                        ALOC0042
      I3=(I3/2)*2+1                                                       ALOC0043
      K1=2*N*NP                                                           ALOC0044
      K2=NACTP*NTP                                                        ALOC0045
      K=K1                                                                ALOC0046
      IF(K2.GT.K1) K=K2                                                   ALOC0047
      IF(IAVG.EQ.1) K=K1                                                  ALOC0048
C --- A AND BINV                                                          ALOC0049
      I4=I3+K                                                             ALOC0050
      I4=(I4/2)*2+1                                                       ALOC0051
C --- NAME                                                                ALOC0052
      I5=I4+NACT                                                          ALOC0053
      I5=(I5/2)*2+1                                                       ALOC0054
C --- P                                                                   ALOC0055
      I6=I5+2*N                                                           ALOC0056
C --- X                                                                   ALOC0057
      I7=I6+2*N                                                           ALOC0058
      MAXN=N                                                              ALOC0059
C --- T                                                                   ALOC0060
      I8=I7+2*NACT                                                        ALOC0061
C --- MA                                                                  ALOC0062
      I9=I8+M*N                                                           ALOC0063
      I9=(I9/2)*2+1                                                       ALOC0064
C --- W                                                                   ALOC0065
      I10=I9+NPLAY*NP                                                     ALOC0066
      I10=(I10/2)*2+1                                                     ALOC0067
C --- D                                                                   ALOC0068
      I11=I10+NPLAY*N                                                     ALOC0069
      I11=(I11/2)*2+1                                                     ALOC0070
```

Reproduced from best available copy.

```
C --- E                                                              ALOC0071
      I12=I11*NPLAY                                                  ALOC0072
      I12=(I12/2)*2+1                                                ALOC0073
C --- MPE                                                            ALOC0074
      I13=I12*NPLAY                                                  ALOC0075
      I13=(I13/2)*2+1                                                ALOC0076
C --- IO                                                             ALOC0077
      I14=I13*N                                                      ALOC0078
      I14=(I14/2)*2+1                                                ALOC0079
C --- II                                                             ALOC0080
      I15=I14*N                                                      ALOC0081
      I15=(I15/2)*2+1                                                ALOC0082
C --- IIYPE                                                          ALOC0083
      I16=I15*NACTMN                                                 ALOC0084
      I16=(I16/2)*2+1                                                ALOC0085
C --- AORE                                                           ALOC0086
      I17=I16*NACTMN                                                 ALOC0087
      I17=(I17/2)*2+1                                                ALOC0088
C --- COMLB                                                          ALOC0089
      I18=I17*2*N                                                    ALOC0090
C --- PLAYLB                                                         ALOC0091
      I19=I13*2*NPLAY                                                ALOC0092
C --- FINMLB                                                         ALOC0093
      I20=I19*2*NACTMN                                               ALOC0094
C --- MAME                                                           ALOC0095
      I21=I20*N                                                      ALOC0096
      I21=(I21/2)*2+1                                                ALOC0097
C --- ALOC                                                           ALOC0098
      I22=I21*N*NPLAY                                                ALOC0099
      I22=(I22/2)*2+1                                                ALOC0100
C --- EXOG                                                           ALOC0101
      I23=I22*N                                                      ALOC0102
      I23=(I23/2)*2+1                                                ALOC0103
C --- IB                                                             ALOC0104
      I24=I23*N                                                      ALOC0105
      I24=(I24/2)*2+1                                                ALOC0106
C --- TT                                                             ALOC0107
      I25=I24*N                                                      ALOC0108
      I25=(I25/2)*2+1                                                ALOC0109
      I26=I25                                                        ALOC0110
      I27=I25                                                        ALOC0111
      I28=I25                                                        ALOC0112
      I29=I25                                                        ALOC0113
      I30=I25                                                        ALOC0114
      I31=I25                                                        ALOC0115
      I32=I25                                                        ALOC0116
      I33=I25                                                        ALOC0117
      I34=I25                                                        ALOC0118
      I35=I25                                                        ALOC0119
      I36=I25                                                        ALOC0120
      I37=I25                                                        ALOC0121
      I38=I25                                                        ALOC0122
      IF(LM.EQ.0) GO TO 50                                           ALOC0123
C --- PP                                                             ALOC0124
      I26=I25*2*LN                                                   ALOC0125
C --- LMA                                                            ALOC0126
      I27=I26*LM*LN                                                  ALOC0127
      I27=(I27/2)*2+1                                                ALOC0128
C --- AP                                                             ALOC0129
      I28=I27*LN                                                     ALOC0130
      I28=(I28/2)*2+1                                                ALOC0131
C --- BP                                                             ALOC0132
      I29=I28*2*LN                                                   ALOC0133
C --- LIO                                                            ALOC0134
      I30=I29*LN                                                     ALOC0135
      I30=(I30/2)*2+1                                                ALOC0136
C --- LII                                                            ALOC0137
      I31=I30*LN                                                     ALOC0138
      I31=(I31/2)*2+1                                                ALOC0139
C --- LMAME                                                          ALOC0140
      I32=I31*LN                                                     ALOC0141
      I32=(I32/2)*2+1                                                ALOC0142
C --- DDL                                                            ALOC0143
      I33=I32*NPLAY                                                  ALOC0144
      I33=(I33/2)*2+1                                                ALOC0145
```

Reproduced from best available copy.

```
C --- PRINV                                                              ALOC0146
      I34=I33+2*LN*LNP                                                  ALOC0147
C --- LIB                                                               ALOC0148
      I35=I34+LN                                                        ALOC0149
      I35=(I35/2)*2+1                                                   ALOC0150
C --- LNAME                                                             ALOC0151
      I36=I35+LN                                                        ALOC0152
      I36=(I36/2)*2+1                                                   ALOC0153
C --- IA                                                                ALOC0154
      I37=I36+LN                                                        ALOC0155
      I37=(I37/2)*2+1                                                   ALOC0156
C --- ALOCLR                                                            ALOC0157
      I38=I37+2*LN                                                      ALOC0158
   50 IF(I38.GT.NSPACE) GO TO 100                                       ALOC0159
      CALL INPUT  (X(I1),X(I2),X(I3),X(I4),X(I5),X(I6),X(I7),          ALOC0160
     * X(I8),X(I9),X(I10),X(I11),X(I12),X(I13),X(I14),X(I15),X(I16),    ALOC0161
     * X(I17),X(I18),X(I19),X(I20),X(I21),X(I22),X(I23),X(I24),X(I25),  ALOC0162
     * X(I26),X(I27),X(I28),X(I29),X(I30),X(I31),X(I32),X(I33),X(I34),  ALOC0163
     * X(I35),X(I36),X(I37))                                            ALOC0164
      GO TO 1                                                           ALOC0165
  100 WRITE(K6,101) I38,NSPACE                                          ALOC0166
  101 FORMAT('0YOU HAVE ATTEMPTED TO ALLOCATE ',I10,  'WORDS WHICH EXCEEALOC0167
     *DS THE',I10,' WORDS AVAILABLE FOR ARRAY STORAGE')                 ALOC0168
      STOP                                                              ALOC0169
 1000 WRITE(K6,1001)                                                    ALOC0170
 1001 FORMAT('0THIS RUN TERMINATES NORMALLY')                           ALOC0171
      STOP                                                              ALOC0172
      END                                                               ALOC0173
      SUBROUTINE INPUT  (WT,H,PINV,A,NAME,P,X,T,MA,b,D,E,XPE,IO,II,ITYPEALOC0174
     * ,AGRE,COMLB,PLAYLB,FIRMLB,NAME,ALOC,EXCG,IB,IT,PP,LMA,AP,        ALOC0175
     * RP,LIO,LII,LNAME,LOL,PRINV,LIB,LNAME,IA,ALOCLB)                  ALOC0176
      DIMENSION WT(1),H(NACT),PINV(NP),A(NACTP,NP),NAME(NACT),P(N),     ALOC0177
     *X(N),T(NACT),MA(NA,1),b(NPLAY,NP),D(NPLAY,1),E(NPLAY),XPE(NPLAY), ALOC0178
     * IO(N),II(N),ITYPE(NACTMN),AGRE(NACTMN),COMLB(1),PLAYLB(NPLAY),    ALOC0179
     * FIRMLB(NACTMN),NAME(N),ALOC(NPLAY,N),EXCG(N),IB(N),IT(N),         ALOC0180
     * LNAME(LN),PRICE(N),LMA(LN,LN),AP(LN),RP(LN),LIO(LN),LII(LN),      ALOC0181
     * LNAME(LN),COL(NPLAY),PRINV(LN,LNP),LIB(LN),IA(LN),ALOCLB(LN)      ALOC0182
      REAL*8 XTYPE(4),XAGRE,XE,XPE                                      ALOC0183
      DOUBLE PRECISION PINV,P,X,T,COMLB,PLAYLB,FIRMLB,PP,RP,PRINV,ALOCLBALOC0184
      DOUBLE PRECISION SUM,X                                            ALOC0185
      COMMON K9,K6,M,NN,MAXI,LM,I99,LMAXI,IMP,ISEL,KPS,ITC              ALOC0186
     *,INDEX,JOUT,IEXS,IDXS1,IDXS2,NS,LINDEX,LJOUT,IAVG                 ALOC0187
      COMMON/DIMS/NACT,NPLAY,LN,NP,LNP,NACTMN,NIP,NACTP                 ALOC0188
      COMMON/LME/XTYPE,XAGRE,XE,XPE                                     ALOC0189
      IF((LM.NE.0).AND.(KPS.EQ.0)) KPS=10                               ALOC0190
      IF(MAXI.EQ.0) MAXI=1000                                           ALOC0191
      IF(LMAXI.EQ.0) LMAXI=1000                                         ALOC0192
      IF(M.EQ.0) M=100                                                  ALOC0193
      IF(LM.EQ.1) WRITE(K6,3)                                           ALOC0194
    3 FORMAT('1THIS IS A GENERAL EQUILIBRIUM RUN  ---  A DESCRIPTION OF ALOC0195
     *THE ECONOMY FOLLOWS')                                             ALOC0196
      IF(LM .NE.0) WRITE(K6,4)                                          ALOC0197
    4 FORMAT('1THIS IS AN ALLOCATION RUN ---'/                          ALOC0198
     *'0A DESCRIPTION OF THE REGIONAL ECONOMY FOLLOWS :')               ALOC0199
      WRITE(K6,5) N,NACT,NPLAY                                          ALOC0200
    5 FORMAT('0NUMBER OF COMMODITIES .....................................ALOC0201
     *..... ',I3/                                                       ALOC0202
     *       ' NUMBER OF PRODUCTION PROCESSES (INCLUDING DISPOSAL ACTIVIALOC0203
     *TIES) .. ',I3/                                                    ALOC0204
     *       ' NUMBER OF CONSUMERS ........................................ALOC0205
     *..... ',I3)                                                       ALOC0206
      IDXS=0                                                            ALOC0207
      IDXS1=0                                                           ALOC0208
      IDXS2=0                                                           ALOC0209
      IF((LM.EQ.0).AND.(NN.EQ.0)) NN=1                                  ALOC0210
      IF(NN.EQ.0) NN=NP                                                 ALOC0211
      IF(LN1.EQ.0) LN1=1                                                ALOC0212
      READ(K5,905)(ALOC(I),I=1,N)                                       ALOC0213
  905 FORMAT(10A8)                                                      ALOC0214
      READ(K5,905)(FIRMLB(I),I=1,NACTMN)                                ALOC0215
      READ(K5,905)(PLAYLB(I),I=1,NPLAY)                                 ALOC0216
   13 WRITE(K6,903)                                                     ALOC0217
  903 FORMAT('0PRODUCTION FUNCTIONS FOR EACH FIRM IN THE ECONOMY'/)     ALOC0218
      WRITE(K6,906)(COMLB(I),I=1,N),XAGRE                               ALOC0219
  906 FORMAT(' ',' FIRM  ',5X,'  TYPE  ',9(4X,A8)/(22X,9(4X,A8)))       ALOC0220
```

Reproduced from best available copy.

```
      WRITE(K6,907)                                                     ALOC0221
  907 FORMAT(' ')                                                       ALOC0222
        DO 14 J=NP,NACT                                                 ALOC0223
      JJ=J-NP                                                           ALOC0224
C --- READ IN THE PRODUCTION SECTOR OF THE ECONOMY                      ALOC0225
      READ(K5,901)ITYPE(JJ),(B(I,J),I=1,N), ACRE(JJ)                   ALOC0226
  901 FORMAT(II,3X,6F10.0/(7F10.0))                                     ALOC0227
      LL=ITYPE(JJ)+1                                                    ALOC0228
      ENG=XTYPE(LL)                                                     ALOC0229
      WRITE(K6,902) FIRMLB(JJ),ENG,(B(I,J),I=1,N), ACRE(JJ)             ALOC0230
  902 FORMAT(' ',A6,4X,A8,9F12.2/(22X,9F12.2))                          ALOC0231
   14 CONTINUE                                                          ALOC0232
      WRITE(K6,803)                                                     ALOC0233
  803 FORMAT('0RESOURCES AVAILABLE TO EACH CONSUMER IN THE ECONOMY'/)   ALOC0234
      WRITE(K6,806)(COMLB(I),I=1,N)                                     ALOC0235
  806 FORMAT(' CONSUMERS',12X,9(4X,A8)/(22X,9(4X,A8)))                  ALOC0236
      WRITE(K6,807)                                                     ALOC0237
  807 FORMAT(' ')                                                       ALOC0238
      NT=N                                                              ALOC0239
      IF(ITO.NE.0) NT=NP                                                ALOC0240
C --- READ IN THE PHYSICAL RESOURCES AVAILABLE TO THE ECONOMY           ALOC0241
        DO 10 I=1,NPLAY                                                 ALOC0242
      READ (K5,100) (A(I,J),J=1,NT)                                     ALOC0243
      WRITE(K6,808) PLAYLB(I),(A(I,J),J=1,NT)                           ALOC0244
  808 FORMAT(' ',A8,13X,9F12.2/(22X,9F12.2))                            ALOC0245
   10 CONTINUE                                                          ALOC0246
  100 FORMAT(7F10.0)                                                    ALOC0247
      WRITE(K6,805)                                                     ALOC0248
  805 FORMAT('0UTILITY FUNCTIONS FOR EACH CONSUMER IN THE ECONOMY'/)    ALOC0249
      WRITE(K6,809)(COMLB(I),I=1,N),XADRE                               ALOC0250
  809 FORMAT(' CONSUMERS',4X,'  TYPE  ',9(4X,A8)/(22X,9(4X,A8)))        ALOC0251
      WRITE(K6,807)                                                     ALOC0252
C --- READ IN THE DEMAND FUNCTION PARAMETERS FOR EACH CONSUMER          ALOC0253
        DO 16 I=1,NPLAY                                                 ALOC0254
      READ(K5,101)  MPE(I),(B(I,J),J=1,N),E(I)                          ALOC0255
  101 FORMAT(II,9X,8F10.0/(7F10.0))                                     ALOC0256
      LL= MPE(I)+1                                                      ALOC0257
      ENG=XTYPE(LL)                                                     ALOC0258
      WRITE(K6,810) PLAYLB(I),ENG,(B(I,J),J=1,N),E(I)                   ALOC0259
  810 FORMAT(' ',A8,5X,A8,9F12.2/(22X,9F12.2))                          ALOC0260
   16 CONTINUE                                                          ALOC0261
C --- READ IN BOUNDARIES                                                ALOC0262
      IF(IBO.EQ.0)  GO TO 150                                           ALOC0263
      READ(K5,140)(IB(I),I=1,N)                                         ALOC0264
  140 FORMAT(16I5)                                                      ALOC0265
      GO TO 160                                                         ALOC0266
  150   DO 155 I=1,N                                                    ALOC0267
      IB(I)=0                                                           ALOC0268
  155 CONTINUE                                                          ALOC0269
  160 WRITE(K6,165)(IB(I),I=1,N)                                        ALOC0270
  165 FORMAT('0UPPER BNDRIES FOR GEN. ECOLIB',16,8I12/(22X,9I12))       ALOC0271
      IF(LM.EQ.0) GO TO 250                                             ALOC0272
      IF(IST.EQ.0) GO TO 170                                            ALOC0273
      READ(K5,140)(IA(I),I=1,LN)                                        ALOC0274
      GO TO 180                                                         ALOC0275
  170   DO 175 I=1,LN                                                   ALOC0276
      IA(I)=I                                                           ALOC0277
  175 CONTINUE                                                          ALOC0278
  180   DO 185 I=1,LN                                                   ALOC0279
      L=IA(I)                                                           ALOC0280
      ALOCLB(I)=COMLB(L)                                                ALOC0281
  185 CONTINUE                                                          ALOC0282
      WRITE(K6,186)(ALOCLB(I),I=1,LN)                                   ALOC0283
  186 FORMAT('0OCCUPY IN ALOC MODEL',9(4X,A8)/(22X,9(4X,A8)))           ALOC0284
      READ(K5,140)(IB(I),I=1,LN)                                        ALOC0285
      WRITE(K6,166)(IB(I),I=1,LN)                                       ALOC0286
  166 FORMAT('0UPPER BNDRIES FOR ALLOCATION  ',16,8I12/(22X,9I12))      ALOC0287
C --- READ IN ABSOLUTE PRICES                                           ALOC0288
      READ(K5,120)(A(I),I=1,LN)                                         ALOC0289
      WRITE(K6,21)(A(I),I=1,LN)                                         ALOC0290
   21 FORMAT('0ABSOLUTE PRICES :',4X,9F12.2/(22X,9F12.2))               ALOC0291
C --- COMPUTE RELATIVE PRICES                                           ALOC0292
      SUM=0.0                                                           ALOC0293
        DO 200 I=1,LN                                                   ALOC0294
      SUM=SUM+AP(I)                                                     ALOC0295
```

Reproduced from best available copy.

```
  200 CONTINUE                                                          ALOC0294
        DO 210 I=1,LN                                                   ALOC0295
      RP(I)=AP(I)/SUM                                                   ALOC0296
  210 CONTINUE                                                          ALOC0297
      WRITE(*6,22)(RP(I),I=1,LN)                                        ALOC0298
   22 FORMAT('ORELATIVE PRICES :',4X,9F12.3/(22X  ,9F12.3))             ALOC0299
C --- READ IN MONEY RESOURCES                                          ALOC0300
      READ(*5,100)(PLE(I),I=1,NPLAY)                                   ALOC0301
      WRITE(*6,23)                                                      ALOC0302
   23 FORMAT('OMONEY RESOURCES AVAILABLE TO EACH CONSUMER:')            ALOC0303
      WRITE(*6,24)(PLAYER(I),I=1,NPLAY)                                 ALOC0304
   24 FORMAT('0',10(4X,A8)/(' ',10(4X,A8)))                             ALOC0305
      WRITE(*6,25)(PLE(I),I=1,NPLAY)                                    ALOC0306
   25 FORMAT('0',10F12.2/(' ',10F12.2))                                 ALOC0307
  250 WRITE(*6,251)                                                     ALOC0308
  251 FORMAT('OTHIS MODEL IS RUN WITH THE FOLLOWING OPTIONS :')         ALOC0309
      IF(LM.*E.0) WRITE(*6,252)                                         ALOC0310
  252 FORMAT('OFOR THE ALLOCATION (OUTER) SOLUTION :')                  ALOC0311
      IF(LM.*E.0) WRITE(*6,253) L*,LM,LN*,LMAXI                         ALOC0312
  253 FORMAT('ONUMBER OF COMMODITIES ..... ',I5/                        ALOC0313
     *'0      ' GRID SIZE ..... ',I5/                                   ALOC0314
     *' .     ' STARTING CORNER ..... ',I5/                             ALOC0315
     *' .     ' MAXIMUM NUMBER OF ITERATIONS ALLOWED .. ',I5)           ALOC0316
      IF(LM.*E.0)    WRITE(*6,257)                                      ALOC0317
  257 FORMAT('OFOR THE GENERAL EQUILIBRIUM (INNER) SOLUTION :')         ALOC0318
      WRITE(*6,253) N,M,NN,MAXI                                         ALOC0319
      WRITE(*6,353) IAVG                                                ALOC0320
  353 FORMAT(' AVERAGING METHOD ..... ',I5)                             ALOC0321
      WRITE(*6,255)                                                     ALOC0322
  255 FORMAT('OOTHER OPTIONS')                                          ALOC0323
      WRITE(*6,256) IBD,ITD,ISEL,EPS                                    ALOC0324
  256 FORMAT('OGEN. EQIL. BOUNDARY OPTION ... ',I5/                     ALOC0325
     *'       ' CONSUMER TAX OPTION ..... ',I5/                        ALOC0326
     *'       ' COMMOD. IN ALLC SELECTION CRIT. ',I5/                  ALOC0327
     *'       ' PRINT OPTION ..... ',I5)                               ALOC0328
      CALL        ALLOCT (WT,N,PINV,A,NAME,P,X,T,MA,M,D,E,MPE,IO,II,ITYPEALOC0329
     * ,AORE,COMLB,PLAYLB,FIRMLB,*NAME,ALLC,EXCD,IB,IT,PW,LMA,AP,        ALOC0330
     * PP,LI),LII,LNAME,DCL,PRINV,LIM,LSAME,IA,SLOCLP)                   ALOC0331
      RETURN                                                            ALOC0332
      END                                                               ALOC0333
      SUBROUTINE ALLOCT (WT,N,PINV,A,SAME,P,X,T,MA,M,D,E,MPE,IO,II,ITYPEALOC0334
     * ,AORE,COMLB,PLAYLB,FIRMLB,NAME,ALOC,EXCD,IB,IT,PP,LMA,AP,        ALOC0335
     * RP,LI),LII,LNAME,DCL,PRINV,LIM,LSAME,IA,SLOCLP)                   ALOC0336
      DIMENSION WT(*),P(*,NACT),PLAY(*,NP),AP(NACTP,NIP),NAME(NACT),P(*),ALOC0337
     *X(N),T(NACT),MA(*,*),M(NPLAY,*),D(NPLAY,*),E(NPLAY),MPE(NPLAY),  ALOC0338
     * IO(N),II(*),ITYPE(NACT*),AORE(NACT*),COMLB(*),PLAYLB(NPLAY),      ALOC0339
     * FIRMLB(NACTP),MAXE(*),ALOC(NPLAY,*),EXCD(*),IB(*),II(*),         ALOC0340
     * LNAME(LN),RP(LN),LMA(LM,LN),AP(LN),RP(LN),LI(LN),LII(LN),        ALOC0341
     * LMAX(LN),DCL(NPLAY),PRINV(LM,LNP),LI(LN),IA(LN),*LOCLP)           ALOC0342
      REAL** XTYPE(6),XAORE,XE,ENG                                      ALOC0343
      DOUBLE PRECISION PINV,P,X,T,COMLB,PLAYLB,FIRMLB,PP,RP,PRINV,ALOCL*ALOC0344
      DOUBLE PRECISION SUM,R                                            ALOC0345
      COMMON KO,K*,M,NN,MAXI,LM,LSN,LMAXI,IBD,ISEL,EPS,ITD              ALOC0346
     *,INDEX,JOUT,IGAS,IUXS1,IUXS2,NS,LINDEX,LJOUT,IAVG                 ALOC0347
      COMMON/DIM/N,NACT,NPLAY,LN,NP,LNP,NACTMA,NIP,NACTP                ALOC0348
      COMMON/LBL/XTYPE,XAORE,XE,ENG                                     ALOC0349
      KKO=0                                                             ALOC0350
      IEND=0                                                            ALOC0351
      NVARY=0                                                           ALOC0352
      IF(LM.EQ.0) GO TO 320                                             ALOC0353
      IF(NN.LE.N) GO TO 260                                             ALOC0354
      NVARY=1                                                           ALOC0355
      NN=1                                                              ALOC0356
C --- INITIALIZE MATRIX PRINV AND LNAME                                 ALOC0357
  260   DO 300 J=1,LN                                                   ALOC0358
        DO 290 I=1,LN                                                   ALOC0359
      PRINV(I,J)=0                                                      ALOC0360
  290 CONTINUE                                                          ALOC0361
      PRINV(J,J)=1.                                                     ALOC0362
      LNAME(J)=J                                                        ALOC0363
      LSAME(J)=J                                                        ALOC0364
  300 CONTINUE                                                          ALOC0365
        DO 310 I=1,LN                                                   ALOC0366
      PRINV(I,LNP)=RP(I)                                                ALOC0367
  310 CONTINUE                                                          ALOC0368
```

Reproduced from best available copy.

```
      LJOUT=LNN                                                          ALOC0371
      LINDEX=1                                                           ALOC0372
      LDEX1=0                                                            ALOC0373
  106 CALL PIVOTA(LMA,LIO,LII,PP,ALOC(LR,LIR)                            ALOC0374
C --- CHECK FOR AN ALLOCATION ON THE BOUNDRY - IF ONE IS FOUND IT MUST HALOC0375
C --- ASSOCIATED WITH A SLACK EQUILIBRIUM PRICE.                         ALOC0376
  330   DO 340 I=1,LN                                                    ALOC0377
      LIP=PP(I)                                                          ALOC0378
      IF(LIP.LE.LIB(I)) GO TO 341                                        ALOC0379
  340 CONTINUE                                                           ALOC0380
      IF(KPS.GE.7) GO TO 349                                             ALOC0381
      WRITE(K6,342) LINDEX,(PP(I),I=1,LN)                                ALOC0382
  342 FORMAT('0',I5,'  ALLOCATION:  ',10F10.4/(21X,10F10.4))             ALOC0383
      GO TO 349                                                          ALOC0384
  341 LIPIV=I                                                            ALOC0385
      LMAME(LJOUT)=I                                                     ALOC0386
        DO 345 J=1,LN                                                    ALOC0387
      IF(J.EQ.LJOUT) GO TO 345                                           ALOC0388
      IF(LMAME(J).EQ.LIPIV) GO TO 347                                    ALOC0389
  345 CONTINUE                                                           ALOC0390
        DO 346 I=1,LN                                                    ALOC0391
      P(I)=0.0                                                           ALOC0392
  346 CONTINUE                                                           ALOC0393
      P(LIPIV)=1.                                                        ALOC0394
      KKO=1                                                              ALOC0395
      IF(KPS.GE.2) GO TO 400                                             ALOC0396
      WRITE(K6,344) LIPIV                                                ALOC0397
  344 FORMAT('0SLACK PRICE ',I5,' BEING INTRODUCED AS AN ACTIVITY')      ALOC0398
      GO TO 400                                                          ALOC0399
  347 LMAME(J)=-1                                                        ALOC0400
      GO TO 502                                                          ALOC0401
C --- COMPUTE PHYSICAL RESOURCES FROM ALLOCATED MONEY RESOURCES          ALOC0402
  349 XM=LN                                                              ALOC0403
        DO 348 I=1,NPLAY                                                 ALOC0404
        DO 348 J=1,N                                                     ALOC0405
      ALOC(I,J)=0.0                                                      ALOC0406
  348 CONTINUE                                                           ALOC0407
        DO 350 I=1,NPLAY                                                 ALOC0408
        DO 350 J=1,LN                                                    ALOC0409
      L=IA(J)                                                            ALOC0410
      ALOC(I,L)=(ALOC(I)*PP(J))/(AP(J)*XN)                               ALOC0411
  350 CONTINUE                                                           ALOC0412
        DO 355 I=1,NPLAY                                                 ALOC0413
        DO 355 J=1,N                                                     ALOC0414
      ALOC(I,J)=ALOC(I,J)+W(I,J)                                         ALOC0415
  355 CONTINUE                                                           ALOC0416
C --- ANY RESOURCE WHICH IS NEGATIVE IS EQUIVILENT TO AN EXOGINOUS DEMAN*ALOC0417
        DO 356 J=1,N                                                     ALOC0418
      EXOG(J)=0.0                                                        ALOC0419
        DO 356 I=1,NPLAY                                                 ALOC0420
      IF(ALOC(I,J).GE.0.0) GO TO 356                                     ALOC0421
      EXOG(J)=EXOG(J)-ALOC(I,J)                                          ALOC0422
      ALOC(I,J)=.0001                                                    ALOC0423
  356 CONTINUE                                                           ALOC0424
      IF(KPS.GE.6) GO TO 360                                             ALOC0425
      WRITE(K6,361)(EXOG(I),I=1,N)                                       ALOC0426
  361 FORMAT('0EXOGINOUS DEMAND : ',9F12.3/(20X,9F12.3))                 ALOC0427
      WRITE(K6,357)                                                      ALOC0428
  357 FORMAT('0NET AVAILABLE RESOURCES IN THE ECONOMY')                  ALOC0429
      WRITE(K6,358)(COMLA(I),I=1,N)                                      ALOC0430
  358 FORMAT('0CONSUMER',8X, 9(A8,4X)/(16X,9(A8,4X)))                    ALOC0431
      WRITE(K6,807)                                                      ALOC0432
  807 FORMAT(' ')                                                        ALOC0433
        DO 365 I=1,NPLAY                                                 ALOC0434
      WRITE(K6,359) PLAYLB(I),(ALOC(I,J),J=1,N)                          ALOC0435
  359 FORMAT(' ',A8,4X,9F12.3/(12X,9F12.3))                              ALOC0436
  365 CONTINUE                                                           ALOC0437
      GO TO 360                                                          ALOC0438
  320   DO 325 J=1,N                                                    ALOC0439
      EXOG(J)=0.0                                                        ALOC0440
        DO 325 I=1,NPLAY                                                 ALOC0441
      ALOC(I,J)=W(I,J)                                                   ALOC0442
  325 CONTINUE                                                           ALOC0443
  360 CALL SCARF(NT,P,FINV,A,NAME,P,X,T,MA,MB,EF,IO,II,ITYP),            ALOC0444
     * AGRE,COMLA,PLAYLB,PLAYER,NAME,ALOC,EXOG,IA,II)                   ALOC0445
```

Reproduced from best available copy.

```
      IF(LM.EQ.0) RETURN                                                ALOC0446
C --- VECTOR P WHICH EXITS FROM SCARF IS THE EQUILIBRIUM PRICE WHICH     ALOC0447
C --- IS ASSICIATED WITH THE CURRENT ALLOCATION.                        ALOC0448
C ---                                                                   ALOC0449
C --- T WILL CONTAIN THE VECTOR P EXPRESSED IN THE E-TH BASIS WHICH     ALOC0450
C --- SHOULD BE ENTERED INTO THE BASIS.                                 ALOC0451
C ---                                                                   ALOC0452
      IV=NN                                                             ALOC0453
      BIG=0.0                                                           ALOC0454
      IF(NVARY.EQ.0) GO TO 371                                          ALOC0455
        DO 370 I=1,N                                                    ALOC0456
      IF(P(I).LT. BIG) GO TO 370                                        ALOC0457
      BIG=P(I)                                                          ALOC0458
      IV=I                                                              ALOC0459
  370 CONTINUE                                                          ALOC0460
  371 NN=IV                                                             ALOC0461
C --- SCALE THE APPROPRIATE LN ELEMENTS OF P TO SUM TO ONE.             ALOC0462
  380 SUM=0.0                                                           ALOC0463
        DO 390 I=1,LN                                                   ALOC0464
      L=IA(I)                                                           ALOC0465
      IF(P(L).GE.0.0) SUM=SUM+P(L)                                      ALOC0466
  390 CONTINUE                                                          ALOC0467
        DO 395 I=1,LN                                                   ALOC0468
      L=IA(I)                                                           ALOC0469
      IF(P(L).LT.0.0) P(I)=0.0                                          ALOC0470
      IF(P(L).GT.0.0) P(I)=P(L)/SUM                                     ALOC0471
  395 CONTINUE                                                          ALOC0472
      WRITE(KA,398)(P(I),I=1,LN)                                        ALOC0473
  398 FORMAT('0FINAL EQUILIB.PRICES',11F10.3/(21X,11F10.3))             ALOC0474
      IF(IEND.EQ.0) GO TO 400                                           ALOC0475
      RETURN                                                            ALOC0476
  400   DO 406 I=1,LN                                                   ALOC0477
      T(I)=0.0                                                          ALOC0478
        DO 403 J=1,LN                                                   ALOC0479
      T(I)=T(I)+PRINV(I,J)*P(J)                                         ALOC0480
  403 CONTINUE                                                          ALOC0481
  406 CONTINUE                                                          ALOC0482
C --- SEARCH FOR THE PIVOT ROW.  LIPIV WILL BE THE ROW                  ALOC0483
  409 R=1000000.                                                        ALOC0484
        DO 420 I=1,LN                                                   ALOC0485
      IF(T(I).LE.0.0) GO TO 420                                         ALOC0486
      IF((PRINV(I,LNP)/T(I)).GE.R) GO TO 420                            ALOC0487
      R=PRINV(I,LNP)/T(I)                                               ALOC0488
      LIPIV=I                                                           ALOC0489
  420 CONTINUE                                                          ALOC0490
      IF(R.LT.900000.) GO TO 425                                        ALOC0491
      WRITE(KA,761)                                                     ALOC0492
  761 FORMAT('0PARTIAL EQUILIBRIUM SOLUTION IS UNBOUNDED')              ALOC0493
      GO TO 600                                                         ALOC0494
C --- RECALCULATE PRINV OR C-F, IE., MATRIX OF ORIGINAL SLACK VECTORS   ALOC0495
C --- IN TERMS OF THE E-TH BASIS.                                       ALOC0496
  425   DO 435 I=1,LN                                                   ALOC0497
      IF(I.EQ.LIPIV) GO TO 435                                          ALOC0498
        DO 430 J=1,LNP                                                  ALOC0499
  430 PRINV(I,J)=PRINV(I,J)-PRINV(LIPIV,J)*T(I)/T(LIPIV)                ALOC0500
  435 CONTINUE                                                          ALOC0501
        DO 440 J=1,LNP                                                  ALOC0502
      PRINV(LIPIV,J)=PRINV(LIPIV,J)/T(LIPIV)                            ALOC0503
  440 CONTINUE                                                          ALOC0504
      LDEXI=LDEX(I+1                                                    ALOC0505
      LNAME(LJOUT)=-1                                                   ALOC0506
      IF(KKO.EQ.1) LNAME(LJOUT)=LIPIV                                   ALOC0507
      KKO=0                                                             ALOC0508
        DO 500 J=1,LN                                                   ALOC0509
      IF(J.EQ.LJOUT) GO TO 500                                          ALOC0510
      IF(LNAME(J).EQ.LIPIV) GO TO 502                                   ALOC0511
  500 CONTINUE                                                          ALOC0512
  502 LNAME(LJOUT)=LNAME(J)                                             ALOC0513
      IF(LIPIV.EQ.LNN) LNAME(LJOUT)=LN                                  ALOC0514
      LJOUT=J                                                           ALOC0515
      IF(KP%.GE.2) GO TO 590                                            ALOC0516
      WRITE(K6,591) LIPIV,LJOUT,(LNAME(I),I=1,LN)                       ALOC0517
  591 FORMAT(2I5,25I5)                                                  ALOC0518
  590 IF(LIPIV.NE.LNN) GO TO 600                                        ALOC0519
      LINDEX=LINDEX+1                                                   ALOC0520
```

Reproduced from best available copy.

```
      IF(LINDEX.EQ.LMAXI) GO TO 600                                     ALOC0521
      GO TO 106                                                         ALOC0522
  600 WRITE(K6,705) LINDEX,LDEX1                                        ALOC0523
  705 FORMAT('0*** THE NUMBER OF ITERATIONS IN THE ALLOCATION SOLUTION IALOC0524
     *S ',I5,' OF WHICH ',I5,' IS/ARE LINEAR PRICES')                   ALOC0525
      WRITE(K6,706) IDX5,IDX52,IDX51                                    ALOC0526
  706 FORMAT('0THE CUMULATIVE NUMBER OF ITERATIONS IN THE GENERAL EQUILIALOC0527
     *BRIUM (I5,2H) SOLUTION IS ',I7/                                   ALOC0528
     * ' OF WHICH ',I7,' ARE LINEAR PROGRAMMING PIVOTS AND ',I7,' INTRODALOC0529
     *UCED ACTIVITIES')                                                 ALOC0530
      LINDEX=-1                                                         ALOC0531
      KPS=0                                                             ALOC0532
      CALL PIVOTA(LMA,LIO,LII,PP,ALOCLB,LIB)                            ALOC0533
      IEND=1                                                            ALOC0534
        DO 853 I=1,LN                                                   ALOC0535
      PP(I)=0.0                                                         ALOC0536
  853 CONTINUE                                                          ALOC0537
      WRITE(K6,850)                                                     ALOC0538
  850 FORMAT('0ALLOCATION  WEIGHT'/)                                    ALOC0539
  851   DO 860 I=1,LN                                                  ALOC0540
      J=LNAME(I)                                                        ALOC0541
      WRITE(K6,857) I,PRINV(J,LNP)                                      ALOC0542
  857 FORMAT(' ',I6,6X,F6.4)                                            ALOC0543
        DO 859 K=1,LN                                                   ALOC0544
      T(I)=LMA(K,I)                                                     ALOC0545
      PP(K)=PP(K)+T(I)*PRINV(J,LNP)                                     ALOC0546
  859 CONTINUE                                                          ALOC0547
  860 CONTINUE                                                          ALOC0548
      XM=LM                                                             ALOC0549
        DO 2010 I=1,LN                                                  ALOC0550
      P(I)=PP(I)/XM                                                     ALOC0551
 2010 CONTINUE                                                          ALOC0552
      WRITE(K6,2020)(PP(I),I=1,LN)                                      ALOC0553
 2020 FORMAT('0EQUILIBRIUM ALLOCATION ',12F9.3/(22X,12F9.3))            ALOC0554
      WRITE(K6,2021)(P(I),I=1,LN)                                       ALOC0555
 2021 FORMAT('0NORMALIZED  ALLOCATION ',12F9.3/(22X,12F9.3))            ALOC0556
      GO TO 999                                                         ALOC0557
      END                                                               ALOC0558
      SUBROUTINE PIVOTA(LMA,LIO,LII,PP,ALOCLB,LIB)                      ALOC0559
      DIMENSION LMA(LN,LM),LIO(LM),LII(LM),PP(LN),ALOCLB(LN),LIB(LN)    ALOC0560
      REAL*8 XIX(6),X*,XF,XG                                            ALOC0561
      DOUBLE PRECISION PP,ALOCLB                                        ALOC0562
      COMMON K5,K6,*,*,*,*,*AXI,LM,L**,LMAXI,IND,ISEL,KPS,ITP          ALOC0563
     *,IXDEX,JOUT,IDX5,IDX51,IDX52,*,*,LINDEX,LDEX1,IAVG               ALOC0564
      COMMON/ZERO/*,*ACT,*PLAY,L**,NP,LNP,*ACTV*,*IP,*ACTP             ALOC0565
      COMMON/ZERO/XIX,XA,XB,XF,XG                                       ALOC0566
      IF(LINDEX.GT.1) GO TO 35                                          ALOC0567
      IF(LINDEX.EQ.-1) GO TO 495                                        ALOC0568
      ISUM=0                                                            ALOC0569
        DO 10 I=1,LM                                                    ALOC0570
      IF(I.EQ.LM) GO TO 5                                               ALOC0571
      ISUM=ISUM+LIB(I)                                                  ALOC0572
    5   DO 10 J=1,LN                                                    ALOC0573
      LMA(I,J)=I+LIB(I)                                                 ALOC0574
   10 CONTINUE                                                          ALOC0575
        DO 15 J=1,LN                                                    ALOC0576
      LMA(LM,J)=LM+2-LN-ISUM                                            ALOC0577
      LMA(J,J)=LMA(J,J)-1                                               ALOC0578
      LIO(J)=J                                                          ALOC0579
      LII(J)=J-1                                                        ALOC0580
   15 CONTINUE                                                          ALOC0581
      LII(1)=LN                                                         ALOC0582
      IF(KPS.GE.2) GO TO 550                                            ALOC0583
      GO TO 490                                                         ALOC0584
   35 CONTINUE                                                          ALOC0585
      JP=LJOUT+1                                                        ALOC0586
      IF(LJOUT.EQ.LM) JP=1                                              ALOC0587
      LOO=LIO(LJOUT)                                                    ALOC0588
      LIO=LII(LJOUT)                                                    ALOC0589
      LOI=LIO(JP)                                                       ALOC0590
      LII=LII(JP)                                                       ALOC0591
      LMA(LOO,LJOUT)=LMA(LOO,LJOUT)+1                                   ALOC0592
      LMA(LII,LJOUT)=LMA(LII,LJOUT)+1                                   ALOC0593
      LMA(LOI,LJOUT)=LMA(LOI,LJOUT)-1                                   ALOC0594
      LMA(LIO,LJOUT)=LMA(LIO,LJOUT)-1                                   ALOC0595
```

Reproduced from best available copy.

```
      LIO(LJOUT)=L01                                                    ALOC0596
      LI1(LJOUT)=L11                                                    ALOC0597
      LIO(JP)=L00                                                       ALOC0598
      LI1(JP)=L10                                                       ALOC0599
      GO TO 550                                                         ALOC0600
  490 WRITE(K6,492)                                                     ALOC0601
  492 FORMAT('0INITIAL ALLOCATION SIMPLEX')                             ALOC0602
      GO TO 500                                                         ALOC0603
  495 IF(KPS.GE.3) GO TO 550                                            ALOC0604
      WRITE(K6,497)                                                     ALOC0605
  497 FORMAT('0EQUILIBRIUM ALLOCATION SIMPLEX')                         ALOC0606
  500   DO 510 I=1,LN                                                   ALOC0607
      WRITE(K6,703) ALOCLB(I),(LMA(I,J),J=1,LA)                         ALOC0608
  703 FORMAT(' ',A8,4X,2016/(13X,2016))                                 ALOC0609
  510 CONTINUE                                                          ALOC0610
  550   DO 560 I=1,LN                                                   ALOC0611
      PP(I)=LMA(I,LJOUT)                                                ALOC0612
  560 CONTINUE                                                          ALOC0613
      RETURN                                                            ALOC0614
      END                                                               ALOC0615
      SUBROUTINE SCARF(NT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,I1,ITYPE, ALOC0616
     * AORE,COMLB,PLAYLB,FIRMLB,NAME,ALOC,EXOG,IB,TT)                   ALOC0617
      DIMENSION NT(1),B(1),NACT(1),BINV(1),P(1),A(NACTP,1),NAME(NACT),P(N),ALOC0618
     *X(N),T(NACT),MA(1,1),W(NPLAY,1),D(NPLAY,1),E(NPLAY),MPE(NPLAY),   ALOC0619
     * IO(N),I1(N),ITYPE(NACTM),AORE(NACTM),COMLB(1),PLAYLB(NPLAY),      ALOC0620
     * FIRMLB(NACTM),NAME(N),ALOC(NPLAY,N),EXOG(N),IB(1),TT(N)          ALOC0621
      REAL*8 XITYPE(4),XAORE,XE,ENG                                     ALOC0622
      DOUBLE PRECISION BINV,P,C,T,COMLB,PLAYLB,FIRMLB                   ALOC0623
      DOUBLE PRECISION B                                                ALOC0624
      COMMON K5,K6,M,NN,MAXI,LM,LAN,LMAXI,IHC,ISEL,KPS,ITC              ALOC0625
     *,INDEX,JOUT,IDXS,IDXS1,IDXS2,NG,LINDEX,LJOUT,IAVG                 ALOC0626
      COMMON/DIM/N,NACT,NPLAY,LN,NP,LNP,NACTMN,NTP,NACTP                ALOC0627
      COMMON/LAL/XITYPE,XAORE,XE,ENG                                    ALOC0628
      INDEX=0                                                           ALOC0629
      INDEX1=0                                                          ALOC0630
      INDEX2=0                                                          ALOC0631
   15   DO 20 J=1,N                                                     ALOC0632
        DO 19 I=1,N                                                     ALOC0633
      B(I,J)=0.                                                         ALOC0634
      BINV(I,J)=0.                                                      ALOC0635
   19 CONTINUE                                                          ALOC0636
      B(J,J)=-1.                                                        ALOC0637
      BINV(J,J)=1.                                                      ALOC0638
      NAME(J)=J                                                         ALOC0639
      NAME(J)=J                                                         ALOC0640
   20 CONTINUE                                                          ALOC0641
      CALL       DEMAND(NT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,I1,ITYPE,ALOC0642
     * AORE,COMLB,PLAYLB,FIRMLB,NAME,ALOC,EXOG,IB,TT)                   ALOC0643
C --- JOUT IS THE COLUMN OF THE PRIMITIVE SET OF PRICES WHOSE ASSOCIATEDALOC0644
C --- SUPPLY OR DEMAND VECTOR IS OUT OF THE BASIS.                      ALOC0645
      JOUT=N                                                            ALOC0646
  106 CALL PIVOT(MA,IO,I1,P,COMLB,IB)                                   ALOC0647
      IP=P(N)+.5                                                        ALOC0648
      IF(IP.LE.IB(N)) GO TO 593                                         ALOC0649
C --- ANY PRICE VECTOR WITH A COMPONENT ON A BOUNDARY WILL BE ASSOCIATEDALOC0650
C --- WITH THE SLACK ACTIVITY WITH A 1 AT THE POSITION OF THE FIRST SUCHALOC0651
        DO 180 J=1,N                                                    ALOC0652
      IP=P(J)+.5                                                        ALOC0653
      IF(IP.LE.IB(J)) GO TO 171                                         ALOC0654
  180 CONTINUE                                                          ALOC0655
C --- FIND THE PRODUCTION PROCESS WITH THE LARGEST PROFIT.  JCOL WILL BEALOC0656
C --- THE PROCESS.                                                      ALOC0657
      PROFIT=-1000000.                                                  ALOC0658
        DO 170 J=NP,NACT                                                ALOC0659
      CALL       SUPPLY(NT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,I1,ITYPE,ALOC0660
     * AORE,COMLB,PLAYLB,FIRMLB,NAME,ALOC,EXOG,IB,TT,J)                 ALOC0661
      SUM=0.                                                            ALOC0662
        DO 169 I=1,N                                                    ALOC0663
  169 SUM=SUM+X(I)*P(I)                                                 ALOC0664
      IF(SUM.LT.PROFIT) GO TO 170                                       ALOC0665
      PROFIT=SUM                                                        ALOC0666
      JCOL=J                                                            ALOC0667
  170 CONTINUE                                                          ALOC0668
      GO TO 172                                                         ALOC0669
C --- THE VECTOR NAME CONTAINS THE PRODUCTION PROCESS NUMBERS WHICH ARE ALOC0670
```

Reproduced from best available copy.

```
C --- ASSOCIATED WITH EACH PRICE VECTOR IN THE CURRENT PRIMITIVE SET.   ALOC0671
C --- IF THE PRICE VECTOR IS ASSOCIATED WITH A DEMAND, THEN A -1 IS     ALOC0672
C --- PLACED IN MAME.                                                   ALOC0673
  171 JCOL=J                                                            ALOC0674
      PROFIT=1.                                                         ALOC0675
  172 MAME(JOUT)=JCOL                                                   ALOC0676
      IF(PROFIT.GE.0.0) GO TO 329                                       ALOC0677
      MAME(JOUT)=-1                                                     ALOC0678
  320 CALL        DEMAND(NT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,I1,ITYPE,ALOC0679
     * AORE,COMLB,PLAYLB,FIRMLB,MAME,ALOC,EXOC,IB,TT)                   ALOC0680
      GO TO 400                                                         ALOC0681
C --- POSITIVE PROFIT SITUATION.                                        ALOC0682
  329   DO 335 J=1,N                                                    ALOC0683
      IF(J.EQ.JOUT) GO TO 335                                           ALOC0684
C --- THIS IS THE TEST TO PREVENT THE SAME ACTIVITY FROM BEING IN THE    ALOC0685
C --- BASIS MORE THAN ONCE.                                             ALOC0686
      IF((JCOL.LE.N).AND.(MAME(J).EQ.JCOL)) GO TO 502                   ALOC0687
      JJ=MAME(J)-N                                                      ALOC0688
      IF(JJ.LE.0) GO TO 335                                             ALOC0689
      IF((ITYPE(JJ).EQ.0).AND.(MAME(J).EQ.JCOL)) GO TO 502              ALOC0690
  335 CONTINUE                                                          ALOC0691
      INDEX1=INDEX1+1                                                   ALOC0692
      CALL        SUPPLY(NT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,I1,ITYPE,ALOC0693
     * AORE,COMLB,PLAYLB,FIRMLB,MAME,ALOC,EXOC,IB,TT,JCOL)              ALOC0694
        DO 340 I=1,N                                                    ALOC0695
      X(I)=-X(I)                                                        ALOC0696
  340 CONTINUE                                                          ALOC0697
  400   DO 406 I=1,N                                                    ALOC0698
C --- T WILL CONTAIN THE VECTOR Y EXPRESSED IN THE CURRENT BASIS WHICH   ALOC0699
C --- SHOULD BE ENTERED INTO THE BASIS.                                 ALOC0700
      T(I)=0.                                                           ALOC0701
        DO 403 J=1,N                                                    ALOC0702
      T(I)=T(I)+BINV(I,J)*X(J)                                          ALOC0703
  403 CONTINUE                                                          ALOC0704
  406 CONTINUE                                                          ALOC0705
      INDEX2=INDEX2+1                                                   ALOC0706
  409 R=1000000.                                                        ALOC0707
C --- SEARCH FOR THE PIVOT ROW.  IPIV WILL BE THE PIVOT ROW.            ALOC0708
        DO 420 I=1,N                                                    ALOC0709
      IF(T(I).LE. .0000001) GO TO 420                                   ALOC0710
      IF(BINV(I,NP)/T(I).GE.R) GO TO 420                                ALOC0711
      R=BINV(I,NP)/T(I)                                                 ALOC0712
      IPIV=I                                                            ALOC0713
  420 CONTINUE                                                          ALOC0714
C --- ECONOMY WILL HAVE AN UNBOUNDED SOLUTION IF ALL THE ELEMENTS OF THE ALOC0715
C --- VECTOR TO BE ENTERED INTO THE BASIS ARE NEGATIVE, OR IF THE RATIO  ALOC0716
C --- IMPLIES THAT THEY ARE ALMOST ALL NEGATIVE.                        ALOC0717
      IF(R.LT.1000000.) GO TO 425                                       ALOC0718
      WRITE(6,761)                                                      ALOC0719
      GO TO 600                                                         ALOC0720
C --- RECALCULATE BINV, THE MATRIX OF ORIGINAL SLACK VECTORS, IN TERMS   ALOC0721
C --- OF THE CURRENT BASIS.                                             ALOC0722
  425   DO 435 I=1,N                                                    ALOC0723
      IF(I.EQ.IPIV) GO TO 435                                           ALOC0724
        DO 430 J=1,NP                                                   ALOC0725
      BINV(I,J)=BINV(I,J)-BINV(IPIV,J)*T(I)/T(IPIV)                     ALOC0726
  430 CONTINUE                                                          ALOC0727
  435 CONTINUE                                                          ALOC0728
C --- RECALCULATE THE PIVOT ROW IN TERMS OF THE CURRENT BASIS.          ALOC0729
        DO 440 J=1,NP                                                   ALOC0730
      BINV(IPIV,J)=BINV(IPIV,J)/T(IPIV)                                 ALOC0731
  440 CONTINUE                                                          ALOC0732
        DO 500 J=1,N                                                    ALOC0733
      IF(J.EQ.JOUT) GO TO 500                                           ALOC0734
      IF(MAME(J).EQ.IPIV) GO TO 502                                     ALOC0735
  500 CONTINUE                                                          ALOC0736
C --- MAME KEEPS TRACK OF WHICH PIVOT ROW IS ASSOCIATED WITH WHICH PRICE ALOC0737
  502 MAME(JOUT)=MAME(J)                                                ALOC0738
      IF(IPIV.EQ.NP) MAME(JOUT)=99                                      ALOC0739
      JOUT=J                                                            ALOC0740
C --- THE ALGORITHM IS COMPLETED WHEN THE PIVOT ROW IS THE NN TH.       ALOC0741
  590 IF(IPIV.EQ.NP) GO TO 600                                          ALOC0742
      INDEX=INDEX+1                                                     ALOC0743
      IF(INDEX.EQ.MAXI) GO TO 600                                       ALOC0744
      GO TO 106                                                         ALOC0745
```

Reproduced from best available copy.

```
C --- THIS IS THE SITUATION WHEN THE TERMINATION IS DUE TO A ZERO PRICE ALOC0746
C --- AT POSITION NN AND HENCE THE PRICE CORNER JOUT IS ASSOCIATED WITH ALOC0747
C --- THE SLACK VECTOR WITH A 1 AT ROW NN                               ALOC0748
  599 NAME(JOUT)=NN                                                     ALOC0749
      NAME(JOUT)=NN                                                     ALOC0750
  600 IDXS=IDXS+INDEX                                                   ALOC0751
      IDXS2=IDXS2+INDEX2                                                ALOC0752
      IDXS1=IDXS1+INDEX1                                                ALOC0753
      IF(KPS.GE.4) GO TO 651                                            ALOC0754
      WRITE(K6,705) INDEX,INDEX2,INDEX1                                 ALOC0755
  705 FORMAT('0*****  THE NUMBER OF ITERATIONS IS ',I5,'  THE NUMBER ALOC0756
     *OF LINEAR PROGRAMMING PIVOTS IS',I5,'  OF WHICH',I5,'  INTRODUCED ALOC0757
     *ACTIVITIES')                                                      ALOC0758
C --- PRINT VECTORS AND LEVELS ASSOCIATED WITH FINAL SIMPLEX.           ALOC0759
      WRITE(K6,623)                                                     ALOC0760
  623 FORMAT('OVECTORS ASSOCIATED WITH EQUILIBRIUM SIMPLEX AND THEIR LEVALOC0761
     *ELS OF OPERATION'/)                                               ALOC0762
      WRITE(K6,621)                                                     ALOC0763
  621 FORMAT('0',2X,'PRICE',3X,'ACTIVITY/DEMAND',3X,'LEVEL',7X,'VECTOR OALOC0764
     *F SUPPLY OR DEMAND CALCULATED ON THE EQUILIBRIUM SIMPLEX'/)        ALOC0765
      WRITE(K6,624)(COMLET(I),I=1,4)                                    ALOC0766
  624 FORMAT(41X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8)          ALOC0767
      WRITE(K6,625)                                                     ALOC0768
  625 FORMAT(' ')                                                       ALOC0769
        DO 627 I=1,NACT                                                 ALOC0770
      T(I)=0.0                                                          ALOC0771
  627 CONTINUE                                                          ALOC0772
        DO 650 K=1,N                                                    ALOC0773
      LL=NAME(K)                                                        ALOC0774
        DO 630 I=1,N                                                    ALOC0775
      P(I)=MA(I,K)                                                      ALOC0776
  630 CONTINUE                                                          ALOC0777
      IF(NAME(K).GT.0)                                                  ALOC0778
     * CALL       SUPPLY(WT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,II,ITYPE,ALOC0779
     * ADRE,COMLB,PLAYLB,FIRMLB,MAME,ALOC,EXOG,IB,TT,NAME(K))           ALOC0780
      IF(NAME(K).LT.0)                                                  ALOC0781
     *CALL        DEMAND(WT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,II,ITYPE,ALOC0782
     * ADRE,COMLB,PLAYLB,FIRMLB,MAME,ALOC,EXOG,IB,TT)                   ALOC0783
      WRITE(K6,622) K,NAME(K),BINV(LL,NP),(X(I),I=1,N)                  ALOC0784
  622 FORMAT(3X,I3,1X,I3,3X,F13.4,MF12.3/(36X,8F12.3))                  ALOC0785
      KK=NAME(K)                                                        ALOC0786
      IF(KK.GT.0) T(KK)=T(KK)+BINV(LL,NP)                               ALOC0787
  650 CONTINUE                                                          ALOC0788
  651 IF(IAV.EQ.1) GO TO 900                                            ALOC0789
      NACTP=2*NACT                                                      ALOC0790
        DO 810 I=1,NACTP                                                ALOC0791
        DO 805 J=1,NTP                                                  ALOC0792
      A(I,J)=0.0                                                        ALOC0793
  805 CONTINUE                                                          ALOC0794
      A(I,NTP)=.0001                                                    ALOC0795
  810 A(I,I)=1.                                                         ALOC0796
        DO 815 J=NP,NACT                                                ALOC0797
      L=J+NACT                                                          ALOC0798
      CALL        SUPPLY(WT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,II,ITYPE,ALOC0799
     * ADRE,COMLB,PLAYLB,FIRMLB,MAME,ALOC,EXOG,IB,TT,J)                 ALOC0800
        DO 815 I=1,N                                                    ALOC0801
      A(I,L)=-X(I)                                                      ALOC0802
  815 CONTINUE                                                          ALOC0803
        DO 820 I=1,N                                                    ALOC0804
      A(I,NTP)=WT(I)                                                    ALOC0805
  820 CONTINUE                                                          ALOC0806
        DO 550 J=1,N                                                    ALOC0807
        DO 825 I=1,N                                                    ALOC0808
      P(I)=MA(I,J)                                                      ALOC0809
  825 CONTINUE                                                          ALOC0810
      CALL        DEMAND(WT,B,BINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,II,ITYPE,ALOC0811
     * ADRE,COMLB,PLAYLB,FIRMLB,MAME,ALOC,EXOG,IB,TT)                   ALOC0812
      L=J+NACT                                                          ALOC0813
        DO 840 I=1,N                                                    ALOC0814
      A(I,L)=X(I)                                                       ALOC0815
  830 CONTINUE                                                          ALOC0816
C --- FOR EACH PRICE ON THE FINAL PRIMITIVE SET, PUT INTO COLUMN L      ALOC0817
C --- STARTING AT THE NTP+1 ROW, THE PROFITS ASSOCIATED WITH EACH PRODUCTALOC0818
C --- ACTIVITY.                                                         ALOC0819
        DO 840 I=NP,NACT                                                ALOC0820
```

Reproduced from best available copy.

```
      A(I,L)=0.                                                          ALOC0821
      CALL        SUPPLY(MT,P,PINV,A,NAME,P,X,T,MA,M,D,E,MPE,IO,II,ITYPE,ALOC0822
     * AORE,COMLB,PLAYLB,FIXLB,NAME,ALOC,EXOG,IB,TT,I)                   ALOC0823
        DO 835 K=1,M                                                     ALOC0824
      A(I,L)=A(I,L)+P(K)*X(K)                                            ALOC0825
835 CONTINUE                                                             ALOC0826
840 CONTINUE                                                             ALOC0827
      A(NACTP,L)=-1.                                                     ALOC0828
850 CONTINUE                                                             ALOC0829
        DO 860 I=1,NACT                                                  ALOC0830
      T(I)=0.                                                            ALOC0831
        DO 855 J=1,N                                                     ALOC0832
      L=J+NACT                                                           ALOC0833
      T(I)=T(I)+A(I,L)                                                   ALOC0834
855 CONTINUE                                                             ALOC0835
860 CONTINUE                                                             ALOC0836
        DO 870 I=1,NACT                                                  ALOC0837
        DO 865 J=1,N                                                     ALOC0838
      L=J+NACT                                                           ALOC0839
      AN=N                                                               ALOC0840
      A(I,L)=A(I,L)-T(I)/(1.+AN)                                         ALOC0841
865 CONTINUE                                                             ALOC0842
870 CONTINUE                                                             ALOC0843
900 INDEX=-1                                                             ALOC0844
      CALL PIVOT(MA,IO,II,P,COMLB,IP)                                    ALOC0845
      CALL        TERM  (MT,M,PINV,A,NAME,P,X,T,MA,M,D,E,MPE,IO,II,ITYPE,ALOC0846
     * AORE,COMLB,PLAYLB,FIXLB,NAME,ALOC,EXOG,IB,TT)                     ALOC0847
      CALL        DEMAND(MT,P,PINV,A,NAME,P,X,T,MA,M,D,E,MPE,IO,II,ITYPE,ALOC0848
     * AORE,COMLB,PLAYLB,FIXLB,NAME,ALOC,EXOG,IB,TT)                     ALOC0849
761 FORMAT(20X,'THE ECONOMY HAS AN UNBOUNDED SOLUTION')                  ALOC0850
701 FORMAT(1X,20F6.2)                                                    ALOC0851
      RETURN                                                             ALOC0852
      END                                                                ALOC0853
      SUBROUTINE PIVOT(MA,IO,II,P,COMLB,IB)                              ALOC0854
      DIMENSION MA(M,N),IO(N),II(M),P(N),COMLB(N),IB(N)                  ALOC0855
      REAL*4 XTYPE(4),XAORE,XE,XG                                        ALOC0856
      DOUBLE PRECISION  P,COMLB                                          ALOC0857
      COMMON KS,MA,M,NO,MAXI,IM,IPX,IMAXI,IND,ISEL,KPS,ITC              ALOC0858
     *,INDEX,JOUT,IO,IO,IE451,IE452,NO,LINDEX,LJOUT,IAVG                 ALOC0859
      COMMON/DIM/N,MACT,NPLAY,IPX,NP,LNP,NACTMX,NIP,NACTP                ALOC0860
      COMMON/LBL/XTYPE,XAORE,XE,XG                                       ALOC0861
      IF(INDEX.GT.1) GO TO 35                                            ALOC0862
      IF(INDEX.EQ.-1) GO TO 495                                          ALOC0863
      ISUM=0                                                             ALOC0864
        DO 10 I=1,M                                                      ALOC0865
      IF(I.EQ.NO) GO TO 5                                                ALOC0866
      ISUM=ISUM+IP(I)                                                    ALOC0867
 5    DO 10 J=1,N                                                        ALOC0868
      MA(I,J)=I+IB(I)                                                    ALOC0869
10 CONTINUE                                                              ALOC0870
        DO 15 J=1,N                                                      ALOC0871
      MA(NO,J)=M+2-N-ISUM                                                ALOC0872
      MA(J,J)=MA(J,J)-1                                                  ALOC0873
      IO(J)=J                                                            ALOC0874
      II(J)=J-1                                                          ALOC0875
15 CONTINUE                                                              ALOC0876
      II(I)=N                                                            ALOC0877
      IF(KPS.GE.2) GO TO 550                                             ALOC0878
      GO TO 490                                                          ALOC0879
35 CONTINUE                                                              ALOC0880
      JP=JOUT+1                                                          ALOC0881
      IF(JOUT.EQ.N) JP=1                                                 ALOC0882
      LOO=IO(JOUT)                                                       ALOC0883
      LIO=II(JOUT)                                                       ALOC0884
      LO1=IO(JP)                                                         ALOC0885
      LII=II(JP)                                                         ALOC0886
      MA(LOO,JOUT)=MA(LOO,JOUT)+1                                        ALOC0887
      MA(LI1,JOUT)=MA(LI1,JOUT)+1                                        ALOC0888
      MA(LO1,JOUT)=MA(LO1,JOUT)-1                                        ALOC0889
      MA(LI1,JOUT)=MA(LI1,JOUT)-1                                        ALOC0890
      IO(JOUT)=LO1                                                       ALOC0891
      II(JOUT)=LII                                                       ALOC0892
      IO(JP)=LOO                                                         ALOC0893
      II(JP)=LIO                                                         ALOC0894
      GO TO 550                                                          ALOC0895
```

Reproduced from best available copy.

```
  490 WRITE(K4,491)                                                     ALOC0896
  491 FORMAT('0*INITIAL PRICE SIMPLEX*/)                                ALOC0897
      GO TO 500                                                         ALOC0898
  495 IF(KPS.GE.4) GO TO 550                                            ALOC0899
      WRITE(K6,496)                                                     ALOC0900
  496 FORMAT('0*EQUILIBRIUM PRICE SIMPLEX*/)                            ALOC0901
  500   DO 510 I=1,N                                                    ALOC0902
      WRITE(K6,703) COMLB(I),(MA(I,J),J=1,N)                            ALOC0903
  703 FORMAT(' ',A8,4X,2016/(13X,2016))                                 ALOC0904
  510 CONTINUE                                                          ALOC0905
  550   DO 560 I=1,N                                                    ALOC0906
      P(I)=AA(I,JOUT)                                                   ALOC0907
  560 CONTINUE                                                          ALOC0908
      RETURN                                                            ALOC0909
      END                                                               ALOC0910
      SUBROUTINE TERM  (WT,P,PINV,A,NAME,P,X,T,MA,W,D,E,MPE,IO,II,ITYPE,ALOC0911
     * AOBE,COMLB,PLAYLB,FIRMLB,NAME,ALOC,FXOG,IP,TT)                   ALOC0912
      DIMENSION W(N),B(N,NACT),PINV(N,NP),A(NACTP,NIP),NAME(NACT),P(N),ALOC0913
     *X(N),T(NACT),MA(N,N),D(NPLAY,NP),E(NPLAY,N),E(NPLAY),MPE(NPLAY),  ALOC0914
     * IO(N),II(N),ITYPE(NACTMP),AOBE(NACTMP),COMLB(N),PLAYLB(NPLAY),   ALOC0915
     * FIRMLB(NACTMP),NAME(N),ALOC(NPLAY,N),FXOG(N),IP(N),TT(N)         ALOC0916
      REAL*8 XTYPE(4),XAOBE,XE,FXG                                      ALOC0917
      DOUBLE PRECISION PINV,P,X,T,COMLB,PLAYLB,FIRMLB                   ALOC0918
      COMMON K5,K6,M,MM,MAXI,LM,LMM,LMAXI,IBC,ISEL,KPS,ITC             ALOC0919
     *,INDEX,JOUT,IEXS,IEAGL,ID(52),NG,LINDEX,LJOUT,IAVG               ALOC0920
      COMMON/BIMAN/NACT,NPLAY,LN,NP,LNP,NACTMP,NIP,NACTP                ALOC0921
      COMMON/LBL/XTYPE,XAOBE,XE,FXG                                     ALOC0922
      NACT=2*NACT                                                       ALOC0923
      AN=N                                                              ALOC0924
      AM=M                                                              ALOC0925
      S=1./AN                                                           ALOC0926
      IF(IAVG.EQ.0) GO TO 1                                             ALOC0927
        DO 4 I=1,N                                                      ALOC0928
      SUM=0.0                                                           ALOC0929
        DO 3 J=1,N                                                      ALOC0930
      R=MA(I,J)                                                         ALOC0931
      SUM=SUM+R*S                                                       ALOC0932
    3 CONTINUE                                                          ALOC0933
      P(I)=SUM/AM                                                       ALOC0934
    4 CONTINUE                                                          ALOC0935
      GO TO 225                                                         ALOC0936
    1   DO 5 I=1,NACT                                                   ALOC0937
      NAME(I)=I                                                         ALOC0938
    5 CONTINUE                                                          ALOC0939
   10   DO 20 J=1,NACTT                                                 ALOC0940
C --- SEARCH FOR A NEGATIVE ELEMENT IN THE BOTTOM ROW.                   ALOC0941
      IF(A(NACTP,J).LE.-.001) GO TO 30                                  ALOC0942
   20 CONTINUE                                                          ALOC0943
      GO TO 199                                                         ALOC0944
C --- THE PIVOT COLUMN IS JPIV                                          ALOC0945
   30 JPIV=J                                                            ALOC0946
      R=10000000.                                                       ALOC0947
        DO 40 I=1,NACT                                                  ALOC0948
      IF(A(I,JPIV).LE..0001) GO TO 40                                   ALOC0949
      IF((A(I,NIP)/A(I,JPIV)).GT.R) GO TO 40                            ALOC0950
      R=A(I,NIP)/A(I,JPIV)                                              ALOC0951
      IPIV=I                                                            ALOC0952
   40 CONTINUE                                                          ALOC0953
      IF(R.GT.9999999.) WRITE(K6,41)                                    ALOC0954
   41 FORMAT('0*UNBOUNDED LINEAR PROGRAM*)                              ALOC0955
      IF(R.GT.9999999.) RETURN                                          ALOC0956
        DO 135 I=1,NACTP                                                ALOC0957
      IF(I.EQ.IPIV) GO TO 135                                           ALOC0958
        DO 130 J=1,NIP                                                  ALOC0959
      IF(J.EQ.JPIV) GO TO 130                                           ALOC0960
      A(I,J)=A(I,J)-A(IPIV,J)*A(I,JPIV)/A(IPIV,JPIV)                    ALOC0961
  130 CONTINUE                                                          ALOC0962
      A(I,JPIV)=.0                                                      ALOC0963
  135 CONTINUE                                                          ALOC0964
        DO 150 J=1,NIP                                                  ALOC0965
      IF(J.EQ.JPIV) GO TO 150                                           ALOC0966
      A(IPIV,J)=A(IPIV,J)/A(IPIV,JPIV)                                  ALOC0967
  150 CONTINUE                                                          ALOC0968
      A(IPIV,JPIV)=1.                                                   ALOC0969
C --- THE VECTOR NAME CONTAINS THE INDEX OF WHICH COLUMNS ARE IN THE     ALOC0970
```

Reproduced from best available copy.

```
C --- BASIS IS SEQUENTIAL ORDER BY UNIT VECTORS.                          ALOC0971
      NAME(IPIV)=JPIV                                                    ALOC0972
      GO TO 10                                                           ALOC0973
  199   DO 200 I=1,N                                                    ALOC0974
      P(I)=0.                                                            ALOC0975
  200 CONTINUE                                                           ALOC0976
        DO 220 J=1,N                                                     ALOC0977
      L=J+NACT                                                           ALOC0978
      R=-1.                                                              ALOC0979
        DO 201 I=1,NACT                                                  ALOC0980
      IF(NAME(I).EQ.L)R=(A(I,NTP)*(AN+1.)/A(NACTP,NTP))-1.               ALOC0981
  201 CONTINUE                                                           ALOC0982
        DO 210 I=1,N                                                     ALOC0983
      S=MA(I,J)                                                          ALOC0984
      P(I)=P(I)+(R*S)/AM                                                 ALOC0985
  210 CONTINUE                                                           ALOC0986
  220 CONTINUE                                                           ALOC0987
  225 IF(KPS.GE.6) RETURN                                                ALOC0988
      WRITE(6,901)                                                       ALOC0989
  901 FORMAT('0EQUILIBRIUM PRICES'/)                                     ALOC0990
      WRITE(K6,910)(COMLB(I),I=1,N)                                      ALOC0991
  910 FORMAT(' ',1X,14(1X,A,))                                           ALOC0992
      WRITE(6,900)(P(I),I=1,N)                                           ALOC0993
  900 FORMAT('0',14F9.3/(' ',14F9.3))                                    ALOC0994
  905 IF(KPS.GE.5) RETURN                                                ALOC0995
      WRITE(6,902)                                                       ALOC0996
  902 FORMAT('0  ACTIVITY',11X,'LEVEL',9X,'PROFIT',10X,'ACTIVITY VECTOR ALOC0997
     *AT EQUILIBRIUM PRICES'/)                                           ALOC0998
      WRITE(K6,915)(COMLB(I),I=1,N)                                      ALOC0999
  915 FORMAT(51X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8,4X,A8)                 ALOC1000
      WRITE(K6,961)                                                      ALOC1001
      IF(IAVG.EQ.1) GO TO 301                                            ALOC1002
        DO 300 J=NP,NACT                                                 ALOC1003
      T(J)=0.0                                                           ALOC1004
  300 CONTINUE                                                           ALOC1005
  301   DO 340 J=NP,NACT                                                 ALOC1006
      JJ=J-N                                                             ALOC1007
      IF(IAVG.EQ.1) GO TO 330                                            ALOC1008
      L=J+NACT                                                           ALOC1009
        DO 310 I=1,NACT                                                  ALOC1010
      IF(NAME(I).EQ.L) T(J)      =(A(I,NTP)*(AN+1.)/A(NACTP,NTP))        ALOC1011
  310 CONTINUE                                                           ALOC1012
  330 PROFIT=0.                                                          ALOC1013
      CALL        SUPPLY(NT,N,NPIV,A,NAME,P,X,I,MA,M,D,F,NPF,IO,II,ITYPE,ALOC1014
     * AORE,COMLB,PLAYLB,FIRMLB,MAMF,ALOC,EXOG,IP,TT,J)                  ALOC1015
        DO 335 I=1,N                                                     ALOC1016
      PROFIT=PROFIT+P(I)*X(I)                                            ALOC1017
  335 CONTINUE                                                           ALOC1018
      WRITE(K6,903) J,FIRMLB(JJ),T(J)      ,PROFIT,(X(K),K=1,N)          ALOC1019
  340 CONTINUE                                                           ALOC1020
  903 FORMAT(' ',I2,3X,A8,3X,F12.3,3X,F12.3,3X,7F12.3/(47X,7F12.3))      ALOC1021
        DO 420 J=NP,NACT                                                 ALOC1022
      CALL        SUPPLY(NT,N,NPIV,A,NAME,P,X,I,MA,M,D,F,NPF,IO,II,ITYPE,ALOC1023
     * AORE,COMLB,PLAYLB,FIRMLB,MAMF,ALOC,EXOG,IP,TT,J)                  ALOC1024
        DO 410 I=1,N                                                     ALOC1025
      WT(I)=WT(I)+T(J)      *X(I)                                        ALOC1026
  410 CONTINUE                                                           ALOC1027
  420 CONTINUE                                                           ALOC1028
      WRITE(K6,920)                                                      ALOC1029
  920 FORMAT('0GENERAL EQUILIBRIUM SOLUTION'/)                           ALOC1030
      WRITE(K6,930)(COMLB(I),I=1,N)                                      ALOC1031
  930 FORMAT(21X,11(2X,A,))                                              ALOC1032
      WRITE(K6,931)(WT(I),I=1,N)                                         ALOC1033
  931 FORMAT('0TOTAL SUPPLY (NET)  ',11F10.3/(21X,11F10.3))              ALOC1034
      WRITE(K6,961)                                                      ALOC1035
  961 FORMAT(' ')                                                        ALOC1036
      RETURN                                                             ALOC1037
      END                                                                ALOC1038
      SUBROUTINE SUPPLY(NT,N,NPIV,A,NAME,P,X,I,MA,M,D,F,NPF,IO,II,ITYPE,ALOC1039
     * AORE,COMLB,PLAYLB,FIRMLB,MAMF,ALOC,EXOG,IP,TT,J)                  ALOC1040
      DIMENSION WT(N),A(M,NACT),NPIV(NP),A(NACTP,NTP),NAME(NACT),P(N),   ALOC1041
     *X(N),T(NACT),MA(M,N),D(PLAY,NP),D(PLAY,N),F(PLAY),NPF(PLAY),       ALOC1042
     * IO(N),II(M),ITYPE(NACT),AORE(NACTP),COMLB(N),PLAYLB(PLAY),        ALOC1043
     * FIRMLB(NACTM),MAMF(II),ALOC(NPLAY,N),EXOG(N),IP(N),TT(N)          ALOC1044
      REAL*8 XTYPE(4),XSORE,XS,END                                       ALOC1045
```

Reproduced from best available copy.

```
      DOUBLE PRECISION BINV,P,X,T,COMLR,PLAYLR,FIRMLR                    ALOC1046
      DOUBLE PRECISION C,RNUM,RDEN                                       ALOC1047
      COMMON K5,K6,N,NN,MAXI,LM,LAN,LMAXI,IBG,ISEL,KPS,ITC              ALOC1048
     *,INDEX,JOUT,IEXS,IDXS1,IDXS2,NG,LENDEA,LJOUT,IAVG                 ALOC1049
      COMMON/DIM2N/NACT,NPLAY,LN,NP,LAP,NACTP,NTP,NACTP                  ALOC1050
      COMMON/LBL/XTYPE,XAERF,XE,FNG                                      ALOC1051
      IF(INDEX.GE.0) GO TO 179                                           ALOC1052
        DO 15 J=1,N                                                      ALOC1053
      BINV(J,NP)=0.                                                      ALOC1054
C --- PLACE INTO THE NPTH COLUMN OF BINV,THE SUM OF INITIAL RESOURCES.   ALOC1055
        DO 14 I=1,NPLAY                                                  ALOC1056
   14 BINV(J,NP)=BINV(J,NP)+ALOC(I,J)                                    ALOC1057
   15 CONTINUE                                                           ALOC1058
      IF(KPS.GE.6) GO TO 16                                              ALOC1059
      WRITE(K6,102)(BINV(J,NP),J=1,N)                                    ALOC1060
  102 FORMAT('OTOTL RESOURCE IN ECON' ,9F12.2/(22X,9F12.2))              ALOC1061
   16 INDEX=1                                                            ALOC1062
C --- WT IS THE VECTOR OF THE TOTAL RESOURCES OF EACH COMMODITY.         ALOC1063
        DO 21 I=1,N                                                      ALOC1064
      WT(I)=BINV(I,NP)                                                   ALOC1065
   21 CONTINUE                                                           ALOC1066
      RETURN                                                             ALOC1067
  179   DO 180 J=1,N                                                    ALOC1068
      X(J)=0.                                                            ALOC1069
  180 CONTINUE                                                           ALOC1070
      SAVE=0.0                                                           ALOC1071
        DO 250 I=1,NPLAY                                                 ALOC1072
      RNUM=0.                                                            ALOC1073
      RDEN=0.                                                            ALOC1074
        DO 190 J=1,N                                                     ALOC1075
      IF(P(J).LE.0.0) GO TO 190                                          ALOC1076
      RNUM=RNUM+P(J)*ALOC(I,J)                                           ALOC1077
      IF(ITD.EQ.0) GO TO 189                                             ALOC1078
      IF(I.EQ.NPLAY) GO TO 185                                           ALOC1079
      Q=RNUM*W(I,NP)                                                     ALOC1080
      RNUM=RNUM-Q                                                        ALOC1081
      SAVE=SAVE+Q                                                        ALOC1082
      GO TO 189                                                          ALOC1083
  185 RNUM=RNUM+SAVE                                                     ALOC1084
  189 IF(MPE(I).EQ.0) RDEN=RDEN+D(I,J)*P(J)                              ALOC1085
      IF(P(J).LE.0.0).AND.(1.-E(I)).LE.0.0) GO TO 190                    ALOC1086
      IF(MPE(I).EQ.2) RDEN=RDEN+D(I,J)*(P(J)**(1.-E(I)))                 ALOC1087
  190 CONTINUE                                                           ALOC1088
      IF(MPE(I).EQ.1) RDEN=1.                                            ALOC1089
      Q=RNUM/RDEN                                                        ALOC1090
        DO 200 J=1,N                                                     ALOC1091
      T(J)=0.                                                            ALOC1092
      IF(MPE(I).EQ.0) T(J)=Q*D(I,J)                                      ALOC1093
      IF(P(J).LE.0.0) GO TO 200                                          ALOC1094
      IF(MPE(I).EQ.1) T(J)=(Q*D(I,J))/(P(J))                             ALOC1095
      IF(MPE(I).EQ.2) T(J)=(Q*D(I,J))/(P(J)**E(I))                       ALOC1096
      X(J)=X(J)+T(J)                                                     ALOC1097
  200 CONTINUE                                                           ALOC1098
      IF(INDEX.GE.0) GO TO 250                                           ALOC1099
      IF(KPS.GE.6) GO TO 250                                             ALOC1100
      WRITE(K6,900) PLAYLR(I),(T(J),J=1,N)                               ALOC1101
  900 FORMAT(' DEMAND OF ',A4,2X,11F10.3/(21X,11F10.3))                  ALOC1102
  250 CONTINUE                                                           ALOC1103
C --- ADD IN EXOGENOUS DEMAND                                            ALOC1104
        DO 325 I=1,N                                                     ALOC1105
      X(I)=X(I)+XD(I)                                                    ALOC1106
  325 CONTINUE                                                           ALOC1107
      IF(KPS.GE.6) RETURN                                                ALOC1108
      IF(INDEX.EQ.-1) WRITE(K6,915)(X(J),J=1,N)                          ALOC1109
  915 FORMAT('OTOTAL DEMAND',8X,11F10.3/(21X,11F10.3))                  ALOC1110
      RETURN                                                             ALOC1111
      END                                                                ALOC1112
      SUBROUTINE SUPPLY(A,F,*,NINV,A,NAME,P,X,T,*,*,*,E,MPE,I,,II,ITYPE,ALOC1113
     * ADRE,COMLR,PLAYLR,FIRMLR,NAME,ALOC,EXOG,I2,I3,I4)                 ALOC1114
      DIMENSION WT(1),MPE(NACT),MPE(N,NP),X(NACTP,NTP),NAME(NACT),P(N),ALOC1115
     *X(N),T(NACT),*,*,*,*,NPLAY,*),*(NPLAY,*),E(NPLAY),*(NPLAY),      ALOC1116
     * I0(N),II(1),I2(NACTMN),I3(NACTMN),ADRE(NACTMN),F(NPLAY),        ALOC1117
     * FIRMLR(NACTMN),NAME(N),ALOC(NPLAY,N),EXOG(N),I0(N),II(1)         ALOC1118
      REAL*8 XTYPE(4),XAERF,XE,FNG                                       ALOC1119
      DOUBLE PRECISION BINV,P,X,T,COMLR,PLAYLR,FIRMLR                    ALOC1120
```

Reproduced from best available copy.

```
      DOUBLE PRECISION PP,PA,SUM                                          ALOC1121
      COMMON KS,K6,M,NN,MAXI,LM,LMA,LMAXI,IBO,ISEL,KPS,IIC               ALOC1122
     *,INDEX,JOUT,IDXS,IDXS1,IDXS2,NG,LINDEX,LJOUT,IAVG                  ALOC1123
      COMMON/DIM/N,NACT,NPLAY,LN,NP,LAP,NACTMA,NTP,NACTP                  ALOC1124
      COMMON/LBL/XTYPE,XAGRE,XE,ENG                                       ALOC1125
      J=IJ                                                                ALOC1126
C --- THE SLACK VECTORS ARE ALWAYS HANDELED WITH ACTIVITY ANALYSIS.        ALOC1127
      IF(J.LE.N) GO TO 100                                                ALOC1128
        DO 10 I=1,N                                                       ALOC1129
      TT(I)=0.0                                                           ALOC1130
   10 CONTINUE                                                            ALOC1131
      JM=J-N                                                              ALOC1132
      JJ=ITYPE(JM)+1                                                      ALOC1133
      GO TO (100,200,300),JJ                                              ALOC1134
C --- ACTIVITY ANALYSIS                                                   ALOC1135
  100   DO 110 I=1,N                                                      ALOC1136
      X(I)=B(I,J)                                                         ALOC1137
  110 CONTINUE                                                            ALOC1138
      RETURN                                                              ALOC1139
C --- COBB-DOUGLAS PRODUCTION FUNCTION                                    ALOC1140
  200   DO 210 I=1,N                                                      ALOC1141
      IF(B(I,J).EQ.-1.) GO TO 205                                         ALOC1142
      GO TO 210                                                           ALOC1143
  205 TT(I)=1.                                                            ALOC1144
  210 CONTINUE                                                            ALOC1145
      PP=1.                                                               ALOC1146
      PA=1.                                                               ALOC1147
        DO 220 K=1,N                                                      ALOC1148
      IF(TT(K).EQ.1.) GO TO 220                                           ALOC1149
      IF((P(K).LE.0.0).OR.(B(K,J).LE.0.0)) GO TO 220                      ALOC1150
      PP=PP*(P(K)**B(K,J))                                                ALOC1151
      PA=PA*(B(K,J)**B(K,J))                                              ALOC1152
  220 CONTINUE                                                            ALOC1153
        DO 250 I=1,N                                                      ALOC1154
      IF(TT(I).EQ.1.) GO TO 250                                           ALOC1155
      X(I)=0.0                                                            ALOC1156
      IF(P(I).LE.0.0) GO TO 250                                           ALOC1157
      X(I)=(PP* B(I,J))/(AGRE(JM)*PA*P(I))                                ALOC1158
      X(I)=-X(I)                                                          ALOC1159
  250 CONTINUE                                                            ALOC1160
        DO 260 I=1,N                                                      ALOC1161
      IF(TT(I).EQ.1.) X(I)=1.                                             ALOC1162
  260 CONTINUE                                                            ALOC1163
      RETURN                                                              ALOC1164
C --- C.E.S. PRODUCTION FUNCTION                                          ALOC1165
  300   DO 310 I=1,N                                                      ALOC1166
      IF(B(I,J).EQ.-1.) GO TO 305                                         ALOC1167
      GO TO 310                                                           ALOC1168
  305 TT(I)=1.                                                            ALOC1169
  310 CONTINUE                                                            ALOC1170
      SUM=0.0                                                             ALOC1171
        DO 320 K=1,N                                                      ALOC1172
      IF(TT(K).EQ.1.) GO TO 320                                           ALOC1173
      IF(P(K).LE.0.0) GO TO 320                                           ALOC1174
      SUM=SUM+(B(K,J))*(P(K)**(1.-AGRE(JM)))                              ALOC1175
  320 CONTINUE                                                            ALOC1176
        DO 350 I=1,N                                                      ALOC1177
      IF(TT(I).EQ.1.) GO TO 350                                           ALOC1178
      X(I)=0.0                                                            ALOC1179
      IF(P(I).LE.0.0) GO TO 350                                           ALOC1180
      X(I)=(B(I,J))/((P(I)**AGRE(JM))*(SUM**(AGRE(JM)/(AGRE(JM)-1.))))    ALOC1181
      X(I)=-X(I)                                                          ALOC1182
  350 CONTINUE                                                            ALOC1183
        DO 360 I=1,N                                                      ALOC1184
      IF(TT(I).EQ.1.) X(I)=1.                                             ALOC1185
  360 CONTINUE                                                            ALOC1186
      RETURN                                                              ALOC1187
      END                                                                 ALOC1188
      BLOCK DATA                                                          ALOC1189
      COMMON/LBL/XTYPE,XAGRE,XE,ENG                                       ALOC1190
      REAL*8 XTYPE(4),XAGRE,XE,ENG                                        ALOC1191
      DATA XTYPE/'ACTIVITY','COBB-DOG',' C.E.S. ','          '/           ALOC1192
      DATA XAGRE/'  A OR E'/                                              ALOC1193
      DATA XE/'ELASTICY'/                                                 ALOC1194
      END                                                                 ALOC1195
```

Reproduced from best available copy.

1 Accession Number	2 Subject Field & Group	SELECTED WATER RESOURCES ABSTRACTS INPUT TRANSACTION FORM
	6CG	

5 Organization

Georgetown University

6 Title

Use of General Equilibrium in Regional Water Resource Planning

10 Author(s)	11 Date	12 Pages	15 Contract Number
Poirier, J. Eugène, S.J. Harrington, John M., Jr.		162	
	16 Project Number	21 Note	
	16110 FIO		

22 Citation

23 Descriptors (Starred First)

*Regional Analysis
*Resource Allocation
*Water Pollution Control
Water Resources
Computer Programs

25 Identifiers (Starred First)

*Algorithm,*Externalities, *Shadow Prices, Regional, Public Goods

27 Abstract

This report presents a methodology for applying general equilibrium analysis to a regional economy. An interaction or trade mechanism is presented which will force a regional economy into equilibrium with the economy in which it is embedded in the sense that the relative prices will be identical in these economies for their common commodities. Specifically one or more consumers in the regional economy is assumed to hold money resources which are allocated to the purchase (sale) in the surrounding economy according to the plan determined by the algorithm presented in this report.

The assumptions required for applying general equilibrium analysis to a region are stated. A technique is presented by which public goods can be treated in a general equilibrium framework. Since one of the conclusions from a general equilibrium application is the optimum supply of any commodity, this technique can determine the optimum supply for a region for such public goods as clean water (air, etc.). There is an equilibrium price for a public good for each consumer and these different prices determine what each consumer should be taxed for the public good.

The results of some numeric computations are presented indicating how the use of the algorithm works with a regional economy which has a public good.

This report was submitted in fulfillment of Project Number 16110 FIO under the sponsorship of the Federal Water Pollution Control Administration.

Abstractor J. Eugène Poirier, S.J.

Institution Georgetown University

WR:102 (REV. OCT. 1968)
WRSIC

SEND TO: WATER RESOURCES SCIENTIFIC INFORMATION CENTER
U.S. DEPARTMENT OF THE INTERIOR
WASHINGTON, D.C. 20240

U.S. GOVERNMENT PRINTING OFFICE 1972 484-483/8R 1-1